AFRL-RI-RS-TR-2016-155

LIFELONG TRANSFER LEARNING FOR HETEROGENEOUS TEAMS OF AGENTS IN SEQUENTIAL DECISION PROCESSES

WASHINGTON STATE UNIVERSITY

JUNE 2016

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2016-155 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
EDWARD VERENICH
Work Unit Manager

/ S /
JULIE BRICHACEK, Chief
Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUNE 2016	FINAL TECHNICAL REPORT	FEB 2014 – FEB 2016

4. TITLE AND SUBTITLE

LIFELONG TRANSFER LEARNING FOR HETEROGENEOUS TEAMS OF AGENTS IN SEQUENTIAL DECISION PROCESSES

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA8750-14-1-0069

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Matthew Taylor, Eric Eaton, and Paul Ruvolo

5d. PROJECT NUMBER
S2MA

5e. TASK NUMBER
TL

5f. WORK UNIT NUMBER
WA

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Washington State University
PO Box 643140
Pullman, WA 99164-3140

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2016-155

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Transferring knowledge from prior experience to a new problem is a key characteristic of human intelligence, enabling us to continually build upon and refine our knowledge. Recent work on transfer in machine learning for autonomous systems has demonstrated that knowledge transfer can improve model performance and accelerate learning. However, current research tends to focus on transfer to a single new problem, showing little consideration for the challenges of transfer learning over consecutive tasks and across diverse domains. Our work develops methods that enable teams of heterogeneous agents to rapidly adapt control and coordination policies to new scenarios, combining lifelong transfer learning and autonomous instruction to support continual transfer among heterogeneous agents and across diverse tasks. We apply these methods to sequential decision-making (SDM) tasks in dynamic environments with simulated and physical robots.

15. SUBJECT TERMS

Sequential decision making, lifelong learning, transfer learning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**EDWARD VERENICH**
U	U	U	UU	61	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This project developed transfer learning methods that enable teams of heterogeneous agents to rapidly adapt control and coordination policies to new scenarios. Our approach uses a combination of **lifelong transfer learning** and **automated instruction** to support continual transfer among heterogeneous agents and across diverse tasks. The resulting system accumulates transferrable knowledge over consecutive tasks, enabling the transfer learning process to improve over time and the system to become increasingly versatile. We apply these methods to sequential decision-making (SDM) tasks in dynamic environments with both simple benchmark tasks and more complex aerial and ground robot tasks.

Our work has produced the following novel contributions:

- **Lifelong accumulation of transferrable knowledge for SDM tasks**: We developed the first general-purpose framework for lifelong reinforcement learning (RL). This general lifelong RL framework supports a wide variety of base RL learning algorithms, provides *theoretical guarantees on performance and convergence*, and has a *computational complexity independent of the number of tasks learned*, ensuring scalability. The system also supports the *reverse transfer* of new knowledge to previously learned models, enabling continual improvement among all agents.
- **Autonomous cross-domain transfer of knowledge**: We developed several approaches for enabling autonomous cross-domain transfer between diverse task domains, including 1) manifold-based methods for transfer learning, and 2) autonomous cross-domain lifelong learning. By projecting diverse tasks into a common space, our approach *supports transfer between domains and models with differing feature and action spaces*, enhancing the system's ability to transfer between diverse SDM tasks. We demonstrated the effectiveness of this approach by showing, for the first time, *cross-domain lifelong learning* between control policies for different dynamical systems (e.g., quadrotors to helicopters to bicycles, etc.)
- **Robust control of aerial and ground robots in the presence of disturbances:** We applied our methods to the problem of learning controllers for robots with novel disturbances in their sensors and actuators. Results show that new robots with novel errors can more rapidly compensate for these imperfections by leveraging lifelong transfer learning.
- **Zero-shot lifelong transfer learning from high-level task descriptions**: We showed that our lifelong learning methods can also incorporate high-level descriptions of the tasks to improve the performance of the lifelong learning process. Most importantly, given a high-level description for a novel task (e.g., the description of a new dynamical system), this approach was able to synthesize a high-performance controller for that new task without observing any training data on that new task.

Improved understanding of automated instruction: Our work has leveraged crowd sourcing as a novel way of collecting data for learning agents. We have shown that naïve users can *provide accurate advice* to an agent learning SDM tasks in multiple settings.
This final report summarizes our most relevant findings over the course of this research grant and briefly highlights remaining open problems.

INTRODUCTION

Humans learn to solve increasingly complex tasks by continually building upon and refining their knowledge. Virtually every aspect of higher-level learning and memory involves this process of knowledge transfer, and it is often credited with enabling *all* learning beyond simple stimulus-response cycles. Recent research on transfer in machine learning seeks to duplicate this notion of reusing knowledge from a set of previously learned *source tasks* to improve learning on a new *target task*. Each task represents a single learning problem, such as determining an action to take in a given situation or recognizing a particular target in imagery. Results show that transfer from previously learned models can improve the performance of new models in a wide variety of problem domains, including handwriting recognition, social network analysis, image classification, maze navigation, simulated robot soccer, and robot control.

However, transfer learning research has yet to enable the rapid and continual *lifelong* learning of complex tasks that occurs in humans. Instead, current research focuses on isolated transfer scenarios to a single target task (known generally as *transfer learning*) or the simultaneously learning of multiple tasks (known as *multi-task learning*), showing little consideration for the intricacies of learning multiple tasks presented in sequence across multiple domains, as occurs in human learning. Such situations occur constantly in operational scenarios where the focus shifts between tasks due to changing mission requirements and/or situational awareness. Autonomous systems have the additional advantage that multiple agents can efficiently share knowledge, enabling transfer learning to provide collective improvement to the system.

Our research has developed transfer learning methods that enable teams of heterogeneous agents to rapidly adapt control and coordination policies to new scenarios. Our approach uses a combination of *lifelong transfer learning* and *automated instruction* to support continual transfer among heterogeneous agents and across diverse tasks, as described in the scenario above. These developed methods have been applied to complex sequential decision-making (SDM) tasks using aerial and ground robots.

METHODS, ASSUMPTIONS, AND PROCEDURES

Our approach was centered on two major thrusts: 1) lifelong transfer learning for sequential decision making, and 2) automated instruction. These methods were developed to support continual transfer among heterogeneous agents and across diverse tasks, as demonstrated in their application to multi-agent search and retrieval using aerial and ground robots. This section summarizes our key developments in each of these areas, with specific results presented in the next section.

Continual Transfer in Lifelong Reinforcement Learning

We developed the first general-purpose lifelong RL framework, which is capable of learning multiple consecutive RL tasks, and autonomously sharing knowledge between the tasks (Figure 1). Our approach is based on upon the Efficient Lifelong Learning Algorithm (ELLA) framework [Ruvolo and Eaton, 2013a, 2013b] (see Appendix A for details) which was originally developed for classification and regression. Our lifelong RL framework can support various base RL algorithms (TD-learning, Q-learning, policy gradients, etc.), providing a mechanism to adapt existing RL methods to the lifelong learning setting. Most notably, we developed PG-ELLA for

lifelong policy gradient learning [Bou Ammar et al., 2014] and GTD-ELLA for **lifelong gradient TD-learning** [Sreenivasan et al, 2014].

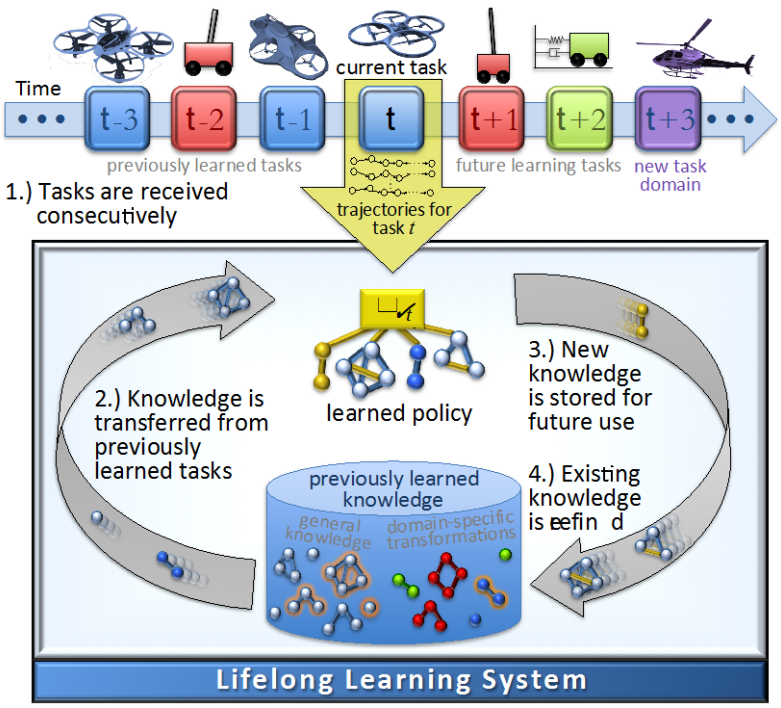


Figure 1: General framework and process for lifelong reinforcement learning, in this case shown with multiple task domains.

Both of these approaches assume a factored representation of the learned knowledge to facilitate transfer (Figure 2). These approaches employ parameterized RL policies, representing the policy for each task as a parameter vector. To facilitate transfer between tasks, our approach learns a shared sparse basis over the parameterized policies, where each basis component effectively represents a chunk of knowledge that captures regularities within the space of policies. The sparse basis is then shared between all tasks, facilitating transfer between the policies as each of them is reconstructed in the shared basis. Given sampled trajectories for a new task, we estimate the policy for that new task using a few steps of RL, and then reconstruct the estimated policy in the shared basis, transferring knowledge through the reconstruction. We show in the next section that this approach is highly effective for lifelong learning of control policies for a variety of dynamical systems and applications to ground vehicle and quadrotor control. Lifelong RL provides a significant jumpstart in initial performance and strong performance gains over learning the new task in isolation through single task learning.

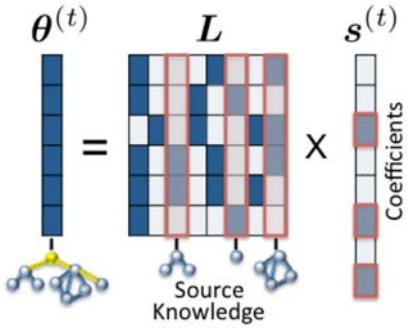


Figure 2: Factored representation of learned model.

We also developed a **fully online version of PG-ELLA** that has reduced computational complexity and exhibits **sublinear regret, thus providing** strong theoretical guarantees [Bou Ammar et al., 2015b] (see Appendix C for details). The fully online variant can also incorporate **safety constraints** into the policy search process, allowing it to produce safe policies. This capability is especially important for practical deployment of lifelong transfer learning, effectively guaranteeing that the policies it produces via transfer will obey given safety

constraints. The form of these safety constraints is particularly versatile, enabling constraints on (for example) allowable states, motion trajectories, and smoothness of transitions.

While all of these methods are highly effective, they are limited to transfer between homogenous agents (e.g., from one quadrotor controller to another quadrotor controller). To remedy this problem, we developed a mechanism to enable **autonomous cross-domain transfer** in lifelong learning systems [Bou Ammar et al., 2015c] (see Appendix D for details). To enable autonomous cross-domain transfer, we added another layer of factorization to the lifelong RL framework (Figure 3). The system learns a single shared basis that underlies all task domains, and then learns a set of projection matrices that specialize that shared basis into a domain-specific basis for each task domain. Next, the domain-specific basis is used to factorize the learned policies for tasks from that domain. This mechanism allows, for the first time, effective transfer between diverse tasks (e.g., between controllers for cart-poles, bicycles, quadrotors, helicopters, etc.). The paper on this approach was *nominated for a best paper award at IJCAI'15*. In addition, to understand the potential mechanisms underlying autonomous cross-domain transfer, we examined the use of **cross-domain manifold alignment** to explain the mapping of state-transitions between pairs of task domains [Bou Ammar et al., 2015a] (see Appendix B for details). This work revealed a strong correlation between the quality of the manifold alignment between two task domains and the quality of the transferred knowledge, providing a potential mechanism for predicting the effectiveness of transfer learning (and thereby avoiding negative transfer).

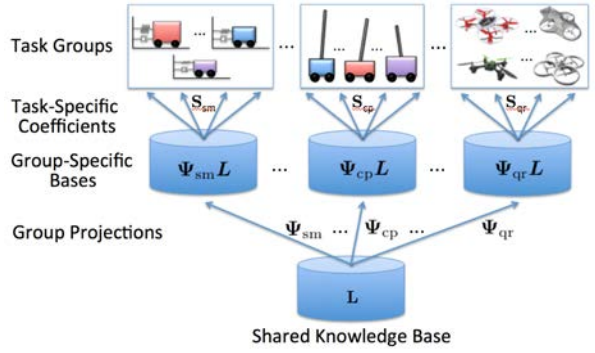


Figure 3: Two-layer factorized framework for autonomous cross-domain transfer between different task domains.

One major issue with this lifelong learning work is that the learner must gain experience in a new task before it can perform transfer. To remedy this situation, we also incorporated high-level task descriptions into the lifelong learning process via coupled dictionary learning [Isele et al, 2016a]. We showed that these high-level task descriptors can improve lifelong learning performance. Most importantly, given the high-level task descriptor for a new task, our approach can synthesize a high-performance controller for that task before gaining any experience interacting with that task, a process known as zero-shot lifelong transfer learning.

Quantifying Negative Transfer

Many existing methods for transfer learning rely exclusively on empirical validation, lacking formal theoretical guarantees. Additionally, the majority of methods require the availability of a (near-) optimal teacher in order to help the student learn. We have addressed these drawbacks by proposing a **new framework for policy advice** [Zhan et al., 2016] (see Appendix E for details). Our framework formally generalizes current single-teacher advice models to the multi-teacher setting. Our novel algorithm also **remedies the need for optimal teachers** by exploiting both the student's and the teacher's knowledge. Even if the teacher is not optimal, a student, using our algorithm, is still capable of acquiring optimal behavior in a task; a property not supported by many state-of-the-art methods, e.g., learning from demonstration. We theoretically and empirically analyze the performance of the proposed method and derive, for the first time, regret bounds quantifying the successfulness of action advice.

These contributions can be summarized as:

- Formally defining multi-teacher advice models,
- introducing novel algorithms leveraging teacher and student knowledge,
- deriving the regret analysis showing reduced sample complexities,
- deriving theoretical guarantees for single teacher advice models, and
- quantifying negative transfer under such advice model.

These theoretical results justify a well-known intuition inherent to advice models: "good teachers help while bad teachers hurt." The results show that students can still achieve optimal behavior when being advised by bad teachers. They, however, pay an extra cost in terms of their learning times or sample complexities, relative to an optimal teacher.

Automated Instruction

In addition to lifelong and transfer learning, we also investigated using on-demand human intelligence to improve learning in SDM tasks. We designed, implemented, and tested a system to interface with Amazon's Mechanical Turk to test if crowd workers are able to accurately provide such advice [de la Cruz Jr. et al., 2015] (see Appendix F for details). Videos from a Pac-Man task were uploaded and the crowd was tested in four distinct scenarios. First, workers were asked to identify mistakes either in a *real-time mode*, where they had only a single opportunity to view agents' actions, or in a *review mode*, where they could pause and rewind a video. Second, workers were asked to either only identify when the agent made a mistake, (i.e., executed suboptimal action), or to identify both when the agent made a mistake and also to suggest an optimal action to execute in its stead. This work is the first research to **establish the crowd's ability to react to mistakes made by an intelligent agent** in real time, and provide accurate guidance on a preferred alternative action. Our work informs the design of future systems that use human intelligence to guide untrained systems through the learning process, without limiting systems to only learn from their mistakes long after they make them.

RESULTS AND DISCUSSION

This section briefly summarizes results from this project. Full details can be found in the cited (publically accessible) papers. The most relevant details are reproduced in appendices A-G)

Lifelong Reinforcement Learning

We evaluated our lifelong RL approaches on the control of a variety of dynamical systems. As a representative algorithm, we first consider PG-ELLA. As shown in Figure 4, PG-ELLA provides significant improvement in performance over learning tasks in isolation using standard policy gradients. As PG-ELLA gains experience with more tasks, its performance improves. We observe the same improvement in the more challenging task of quadrotor control (Figure 5).

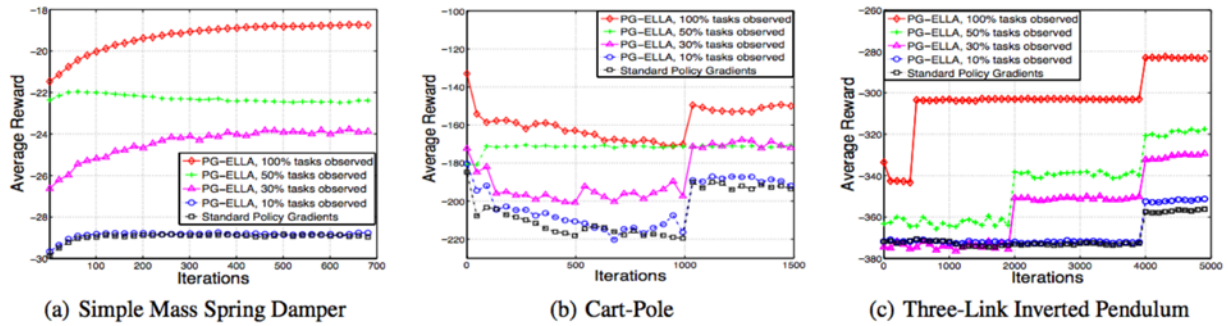


Figure 4: The performance of PG-ELLA vs standard policy gradients (eNAC) on benchmark dynamical systems.

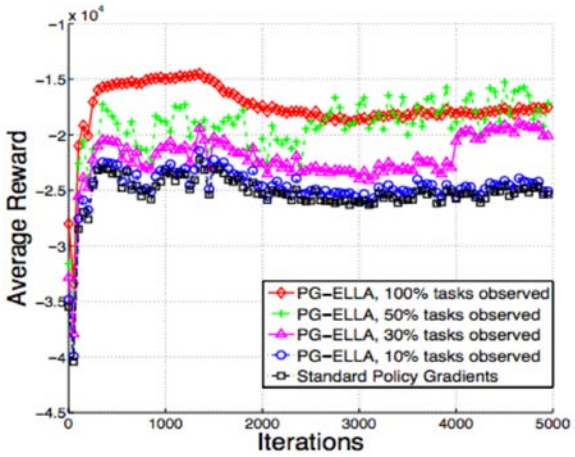


Figure 5: Performance of lifelong RL on quadrotor control.

As shown in Figure 6 and Figure 7, the fully-online safe lifelong learner produces policies that can outperform PG-ELLA on the control of dynamical systems. Figure 6 (c-d) and Figure 8 also show that the policies produced by the learner always obey the given safety constraints.

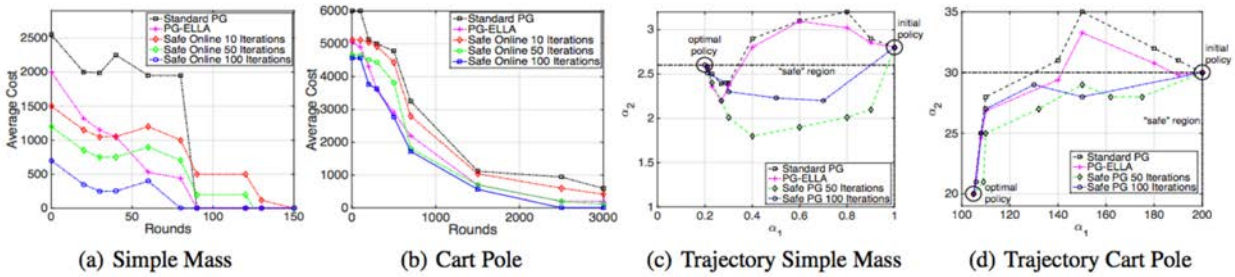


Figure 6: Results of Safe Online Lifelong RL on benchmark simple mass and cart-pole systems. Figures (a) and (b) depict performance in lifelong learning scenarios over consecutive unconstrained tasks, showing that our approach outperforms standard PG and PG-ELLA. Figures (c) and (d) examine the ability of these method to abide by safety constraints on sample constrained tasks, depicting two dimensions of the policy space (α_1 vs. α_2) and demonstrating that our approach abides by the constraints (the dashed black region).

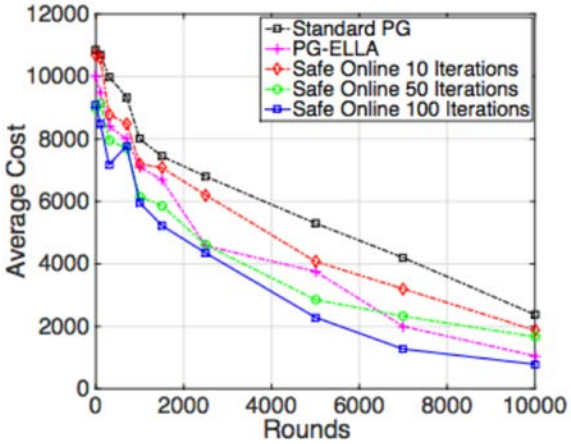


Figure 7: Performance of Safe Online Lifelong RL on quadrotor control.

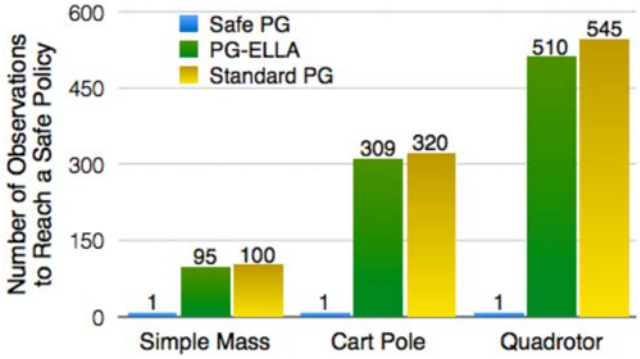


Figure 8: Average number of task observations before acquiring policy parameters that abide by the constraints, showing that Safe Online Lifelong RL immediately projects policies to safe regions.

We showed that incorporating high-level task descriptions improves the performance of multi-task learning (MTL) and lifelong learning (Figure 9), and that our approach also supports zero-shot transfer to new tasks given only their high-level task description (Figure 10).

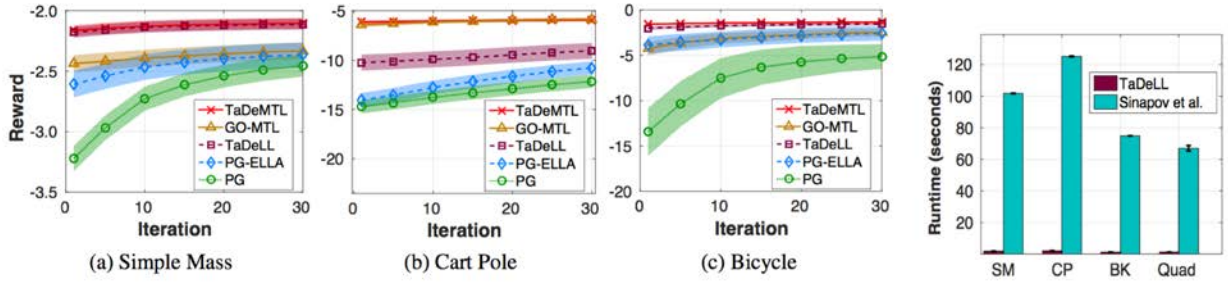


Figure 9: Performance of multi-task (solid lines), lifelong (dashed), and single-task learning (dotted) on benchmark dynamical systems. The TaDeMTL and TaDeLL algorithms represent MTL and lifelong learning, respectively, with high-level task descriptors. The rightmost figure compares the run-time performance of lifelong learning with task descriptors against an alternative (computationally expensive) method for reinforcement learning with task descriptors.

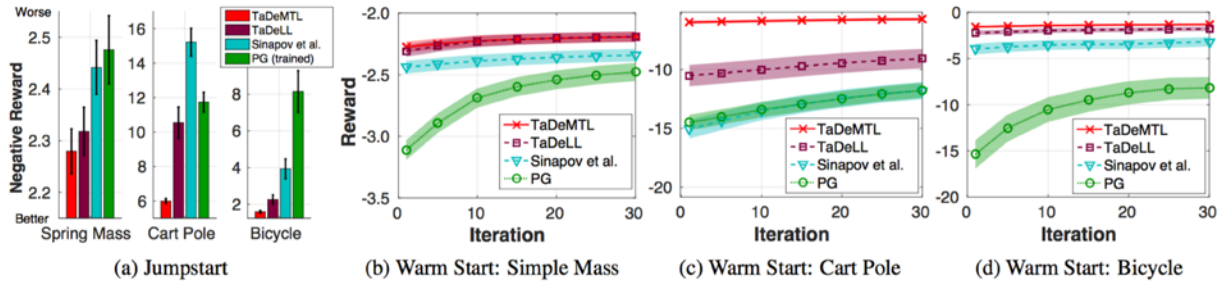


Figure 10: Zero-shot transfer learning to new tasks. Figure (a) shows the initial "jumpstart" improvement on each task domain; Figures (b)–(d) depict the result of using zero-shot policies as warm start initializations for PG.

In addition to the simple simulation domains discussed above, we also investigated lifelong learning on complex simulators and robots [Isele et al., 2016b]. This work explicitly frames lifelong learning as that of disturbance rejection: given that all robots are slightly different, learning on a new robot should be faster after learning on a number of similar robots. In these experiments we show that our methods do generalize to real-world domains. This work (see Figure 11) shows that lifelong learning did successfully improve learning in simulated turtlebots, simulated AR-Drones, and physical turtlebots.

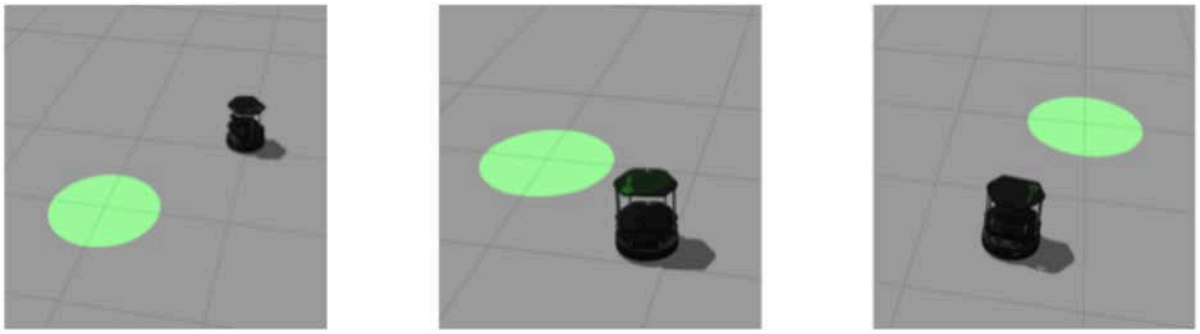


Figure 11: Example setting where a simulated turtlebots with different imperfections must learn to reach different target areas.

To evaluate cross-domain lifelong learning, we interleaved tasks from six different domains (simple mass, double mass, cart pole, double cart pole, bicycle, helicopter), and then evaluated learning performance on each task domain. Figure 12 shows that cross-domain transfer provides a substantial benefit to lifelong learning, significantly increasing performance above learning (via PG-ELLA) on a single domain of tasks. We also evaluated the performance of cross-domain transfer learning to a novel task domain after having observed tasks from other task domains. We chose the most difficult task domain (helicopter) as the novel domain. Figure 13 shows the performance on the helicopter domain, after learning on the other five task domains. These results demonstrate that cross-domain transfer provides a significant gain in performance on the novel task domain. Figure 14 shows the difference in policy reconstruction quality (an indicator of the success of transfer learning) versus the Procrustes measure (a measure of manifold alignment quality). The high correlation between these two quantities suggests that it may be possible to examine the manifold alignment quality for the transitions underlying each domain in order to assess whether or not transfer will succeed.

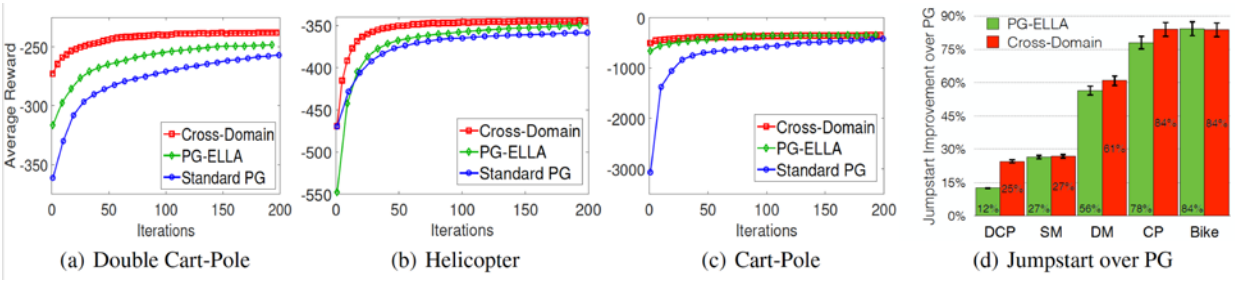


Figure 12: Performance of cross-domain lifelong learning after interleaved training over multiple task domains. Figures (a) and (b) depict task domains where cross-domain transfer has a significant impact, showing that our approach outperforms standard PG and PG-ELLA. Figure (c) demonstrates that even when a domain benefits less from cross-domain transfer, our approach still achieves equivalent performance to PG-ELLA. Figure (d) depicts the average improvement in initial task performance over PG from transfer.

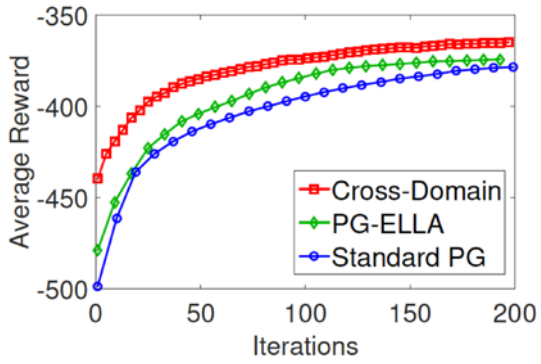


Figure 13: Performance of cross-domain transfer on a novel task domain (helicopter) after lifelong learning on other domains.

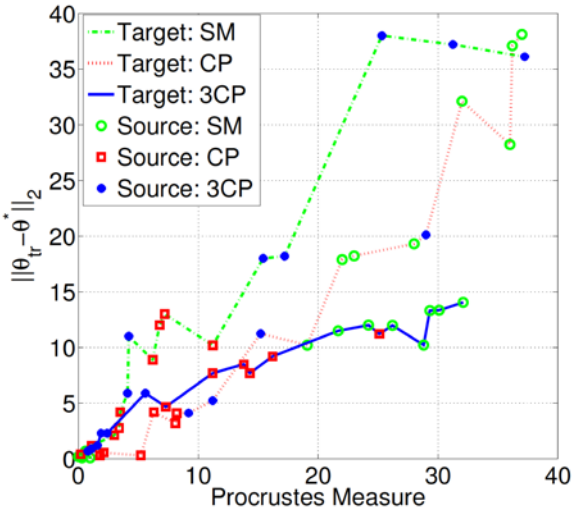


Figure 14: Correlation between manifold alignment quality (Procrustes metric) and quality of the transferred knowledge for cross-domain transfer between simple mass (SM), cart pole (CP), and three-link cart pole (3CP) systems.

Quantifying Negative Transfer

In our recent paper [Zhan et al., 2016] we show that it is possible to quantify negative transfer when using a class of advice algorithms. In particular, we introduce a *regret ratio* that defines how good a teacher is, and then use this ratio to bound how much this teacher can help a student learn. In additional to theoretical work, a set of experiments show that our algorithms work in practice with a variety of teachers (see Figure 15). This work opens the possibility of future studies with other classes of advice algorithms, as well as allows us to better understand how and why negative transfer occurs.

The results of this work are as follows. First, it quantifies the occurrence of negative transfer in action advice models, shedding light on the failure modes of these methods. Second, we show that high-quality transfer knowledge may still cause negative transfer when the target algorithm is able to outperform the source knowledge. Third, it is important for the researchers to determine whether or not to transfer the expert knowledge because evaluation of the transfer knowledge is expensive (it is equivalent to evaluating the teacher policy in the target MDP).

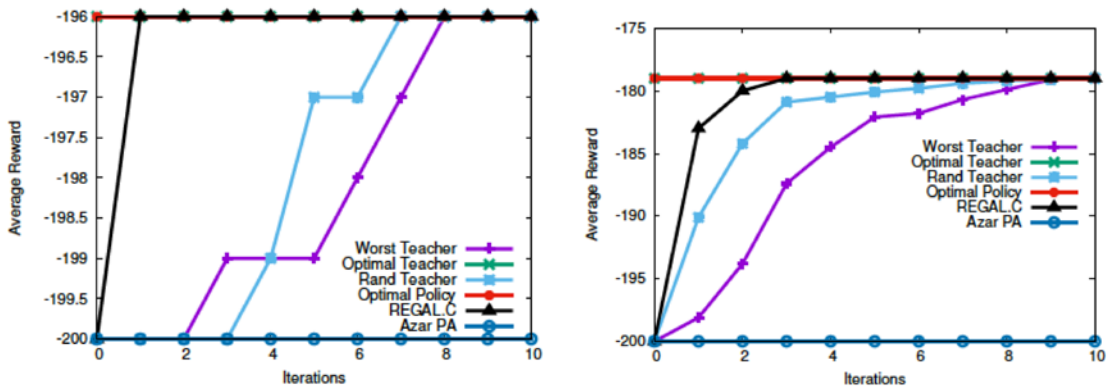


Figure 15: Experiments on "Combination Lock," a simple benchmark task (Left) and "Block Dude," a complex sequential decision task (Right) , show that our method can benefit not only from an optimal teacher, but also from a random teacher and a teacher that always suggests the worst actions.

Automated Instruction

To generate four videos used in the user study, we recorded Pac-Man (see Figure 16, Left) being controlled by a human who intentionally made different types of mistakes. Then, we picked 10-14 seconds that contained one (and only one) mistake. We then created 16 Human Intelligence Tasks (HITs) on Amazon's Mechanical Turk (4 videos in each of the 4 settings). 30 unique workers performed each of the 16 HITs.

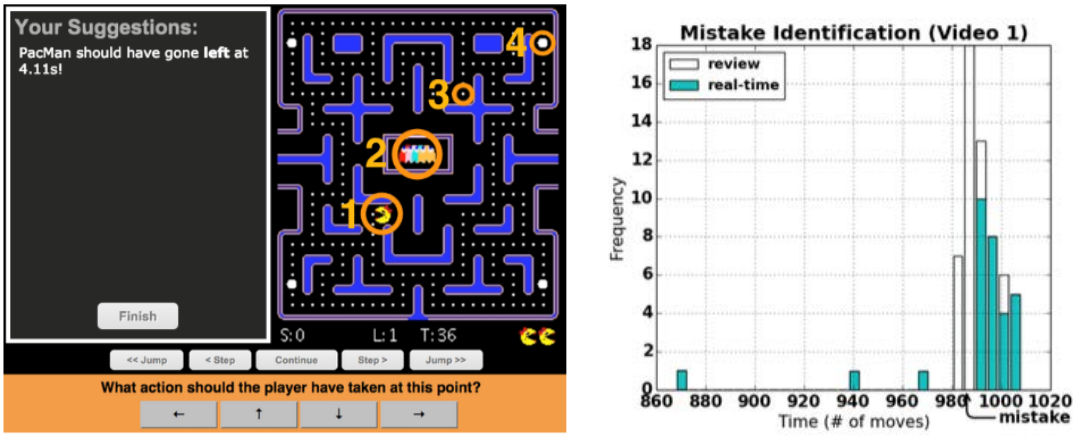


Figure 16: LEFT: This screenshot shows the web interface of the user study with game layout and components of the Pac-Man game: 1) Pac-Man, 2) 4 ghosts, 3) Pills, and 4) Power Pills. RIGHT: A histogram of the distribution of workers' suggestions.

Results from this user study supported multiple hypotheses. First, the crowd can collectively identify the correct point at which an error occurs with over 91% accuracy (see Figure 16, Right). Second, we demonstrate that not only can this mistake identification be done in real time (e.g., as would be done with a live video of a robot learning) with a mean latency of just 0.39

seconds, workers are also able to successfully identify what the optimal move should have been. Third, we compare the crowd's performance in this real-time setting with an offline "review" setting where game playback can be controlled and replayed. If additional time is available (e.g., a video of a robot performing a task that can be watched multiple times), any mistakes can be better estimated: our data show a mean latency of 0.15 seconds.
The contributions in this research are to:

- Present and formalize the idea that on-line crowds can provide assistance to learning agents in real-time, and as the need arises, to improve performance.
- Demonstrate that crowd workers from readily accessible platforms such as Amazon Mechanical Turk can respond quickly and accurately enough to provide just-in-time feedback.
- Show that workers can also improve their accuracy in post hoc review settings to yield better performance in future situations.
- Discuss the types and sources of errors from the crowd observed in the 16 studies that will be critical for successful integration of crowd feedback and reinforcement learning.

Future work will include fully integrating knowledge from the crowd with autonomous learning. We expect such knowledge will allow agents to more quickly learn SDM tasks, with relatively little overhead.

CONCLUSION

This work has substantially advanced the field of lifelong learning for sequential decision tasks, resulting in more than 10 peer reviewed papers, including one that was nominated for a best paper award at IJCAI'15. We have developed and improved multiple algorithms that allow simulated and physical robots to learn a sequence of tasks, continually accumulating knowledge, while improving performance on past tasks. This sequence can be formed from a set of tasks from the same domain, or can be formed from multiple domains (e.g., cross-domain transfer). Additionally, human knowledge can be collected from non-expert users, which can potentially improve the performance both of transfer learning and of base learning algorithms.

While this work has been largely successful, it has also raised multiple exciting open questions for future work.

- How can lifelong learning methods be extended beyond linear parametric models to other types of base learning algorithms (e.g., kernel methods, deep learning, etc.) to produce **more powerful lifelong learning algorithms**?
- How and under what conditions can we **autonomously transfer knowledge between diverse tasks,** including across agents situated in radically different environments?
- Can the use of **hierarchical knowledge** in lifelong learning allow us to scale to large, more complex problems?
- How can we incorporate **high-level task descriptions** or partial information into transfer learning to synthesize models for novel, complex tasks?
- How can **safe reinforcement learning** best be applied to physical robots in order to guarantee policies that obey required constraints?
- Can **negative transfer** be quickly identified and avoided?
- How can **human advice** best be incorporated into the lifelong learning process?

REFERENCES

Bou Ammar et al., 2014] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. Online Multi-Task Learning for Policy Gradient Methods. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, June 2014.

[Bou Ammar et al., 2015a] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, January 2015.

[Bou Ammar et al., 2015b] Haitham Bou Ammar, Rasul Tutunov, and Eric Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, July 2015.

[Bou Ammar et al., 2015c] Haitham Bou Ammar, Eric Eaton, Jose Marcio Luna, and Paul Ruvolo. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-15)*, July 2015.

[de la Cruz Jr. et al., 2015] Gabriel V. de la Cruz Jr., Bei Peng, Walter S. Lasecki, and Matthew E. Taylor. Towards Integrating Real-Time Crowd Advice with Reinforcement Learning. In *Proc. of the 20th ACM Conf. on Intelligent User Interfaces (IUI-15)*, March 2015. [Poster]

[Isele et al., 2016a] David Isele, Mohammad Rostrami, and Eric Eaton. Using Task Features for Zero-Shot Knowledge Transfer in Lifelong Learning. *To appear in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-16)*, July 2016.

[Isele et al., 2016b] David Isele, José Marcio Luna, Eric Eaton, Gabriel de la Cruz, James Irwin, Brandon Kallaher, and Matthew E. Taylor. Lifelong Learning for Disturbance Rejection on Mobile Robots. *Submitted to the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-16)*, October 2016.

[Sreenivasan et al., 2014] Vishnu Purushothaman Sreenivasan, Haitham Bou Ammar, and Eric Eaton. Online Multi-Task Gradient Temporal-Difference Learning. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, July 2014. [Student Abstract]

[Ruvolo and Eaton, 2013a] Paul Ruvolo and Eric Eaton. ELLA: An Efficient Lifelong Learning Algorithm. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, June 2013.

Ruvolo and Eaton, 2013b] Paul Ruvolo and Eric Eaton. Active Task Selection for Lifelong Machine Learning. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI-13)*, July 2013.

[Zhan et al., 2016] Yusen Zhan, Haitham Bou Ammar, and Matthew E. Taylor. Theoretically-Grounded Policy Advice from Multiple Teachers in Reinforcement Learning Settings with Applications to Negative Transfer. In Proceedings of the 25th International Conference on Artificial Intelligence (IJCAI-16), July 2016.

The following appendices provide additional technical detail regarding the most salient contributions referenced in the above final report.

Appendix A: Ruvolo and Eaton, 2013a

ELLA: An Efficient Lifelong Learning Algorithm

Paul Ruvolo PRUVOLO@CS.BRYNMAWR.EDU
Eric Eaton EEATON@CS.BRYNMAWR.EDU
Bryn Mawr College, Computer Science Department, 101 North Merion Avenue, Bryn Mawr, PA 19010 USA

Abstract

The problem of learning multiple consecutive tasks, known as *lifelong learning*, is of great importance to the creation of intelligent, general-purpose, and flexible machines. In this paper, we develop a method for online multi-task learning in the lifelong learning setting. The proposed Efficient Lifelong Learning Algorithm (ELLA) maintains a sparsely shared basis for all task models, transfers knowledge from the basis to learn each new task, and refines the basis over time to maximize performance across all tasks. We show that ELLA has strong connections to both online dictionary learning for sparse coding and state-of-the-art batch multi-task learning methods, and provide robust theoretical performance guarantees. We show empirically that ELLA yields nearly identical performance to batch multi-task learning while learning tasks sequentially in three orders of magnitude (over 1,000x) less time.

1. Introduction

Versatile learning systems must be capable of efficiently and continually acquiring knowledge over a series of prediction tasks. In such a lifelong learning setting, the agent receives tasks sequentially. At any time, the agent may be asked to solve a problem from any previous task, and so must maximize its performance across all learned tasks at each step. When the solutions to these tasks are related through some underlying structure, the agent may share knowledge between tasks to improve learning performance, as explored in both transfer and multi-task learning.

Despite this commonality, current algorithms for transfer and multi-task learning are insufficient for lifelong learning. Transfer learning focuses on efficiently modeling a new target task by leveraging solutions to previously learned source tasks, without considering potential improvements to the source task models. In contrast, multi-task learning (MTL) focuses on maximizing performance across *all* tasks through shared knowledge, at potentially high computational cost. Lifelong learning includes elements of both paradigms, focusing on efficiently learning each consecutive task by building upon previous knowledge while optimizing performance across all tasks. In particular, lifelong learning incorporates the notion of *reverse transfer*, in which learning subsequent tasks can improve the performance of previously learned task models. Lifelong learning could also be considered as online MTL.

In this paper, we develop an Efficient Lifelong Learning Algorithm (ELLA) that incorporates aspects of both transfer and multi-task learning. ELLA learns and maintains a library of latent model components as a shared basis for all task models, supporting soft task grouping and overlap (Kumar & Daumé III, 2012). As each new task arrives, ELLA transfers knowledge through the shared basis to learn the new model, and refines the basis with knowledge from the new task. By refining the basis over time, newly acquired knowledge is integrated into existing basis vectors, thereby improving previously learned task models. This process is computationally efficient, and we provide robust theoretical guarantees on ELLA's performance and convergence. We evaluate ELLA on three challenging multi-task data sets: land mine detection, facial expression recognition, and student exam score prediction. Our results show that ELLA achieves nearly identical performance to batch MTL with three orders of magnitude (over 1,000x) speedup in learning time. We also compare ELLA to a current method for online MTL (Saha et al., 2011), and find that ELLA has both lower computational cost and higher performance.

2. Related Work

Early work on lifelong learning focused on sharing distance metrics using task clustering (Thrun & O'Sullivan, 1996), and transferring invariances in neu-

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

ral networks (Thrun, 1996). Lifelong learning has also been explored for reinforcement learning (Ring, 1997; Sutton et al., 2007) and learning by reading (Carlson et al., 2010). In contrast, ELLA is a general algorithm that supports different base learners to learn continually, framed in the context of current MTL methods.

Recently, MTL research has considered the use of a shared basis for all task models to improve learning over a set of tasks. Several formulations of this idea have been proposed, including a probabilistic framework (Zhang et al., 2008) and a non-parametric Bayesian method that automatically selects the number of bases (Rai & Daumé III, 2010). These methods assume that each model is represented as a parameter vector that is a linear combination of these bases. By using a common basis, these approaches share information between learning tasks and account for task relatedness as the models are learned in tandem with the basis. The GO-MTL algorithm (Kumar & Daumé III, 2012) also uses a sparsely shared basis for multi-task learning, with the advantage that it automatically learns (potentially) overlapping groups of tasks to maximize knowledge transfer. We employ this rich model of underlying task structure as the starting point for developing ELLA.

Few papers have focused on the development of very computationally efficient methods for MTL. Simm et al. (2011) present a model for learning multiple tasks that is efficient in the case when the number of tasks is very large. However, their approach suffers from significant drawbacks in comparison with ELLA: (1) their approach is not an online algorithm, limiting its use in the lifelong learning setting, and (2) their underlying model of shared task structure is significantly less flexible than our model. Another approach, OMTL by Saha et al. (2011), is designed to provide efficient performance when instances and new tasks arrive incrementally. However, OMTL is only applicable to classification tasks (not regression) and relies on perceptron learning, which we found to perform poorly in comparison to other base learners (see Section 4).

3. Approach

We begin by describing the lifelong learning problem and why lifelong learning algorithms must be able to efficiently learn new tasks and incorporate new training data from previous tasks. Then, we introduce ELLA, which efficiently handles both of these operations through a two-stage optimization procedure. We show that ELLA encompasses the problem of online dictionary learning for sparse coding as a special case, and finish by proving robust convergence guarantees.

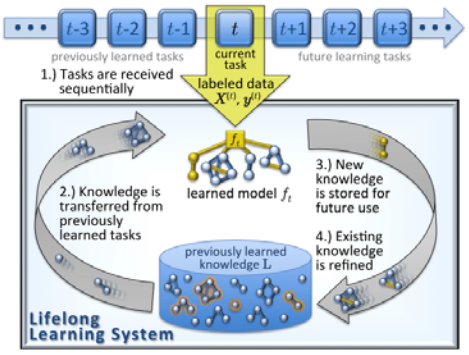


Figure 1. An illustration of the lifelong learning process.

This paper uses the following conventions: matrices are denoted by bold uppercase letters, vectors are denoted by bold lowercase letters, scalars are denoted by normal lowercase letters, and sets are denoted using script typeface (e.g., $\mathcal{A}$). Parenthetical superscripts denote quantities related to a particular task (e.g., matrix $\mathbf{A}^{(t)}$ and vector $\mathbf{v}^{(t)}$ are related to task t).

3.1. The Lifelong Learning Problem

A lifelong learning agent (Figure 1) faces a series of supervised learning tasks $\mathcal{Z}^{(1)}, \mathcal{Z}^{(2)}, \ldots, \mathcal{Z}^{(T_{\max})}$, where each task $\mathcal{Z}^{(t)} = \left(\hat{f}^{(t)}, \mathbf{X}^{(t)}, \mathbf{y}^{(t)}\right)$ is defined by a (hidden) mapping $\hat{f}^{(t)} : \mathcal{X}^{(t)} \mapsto \mathcal{Y}^{(t)}$ from an instance space $\mathcal{X}^{(t)} \subseteq \mathbb{R}^d$ to a set of labels $\mathcal{Y}^{(t)}$ (typically $\mathcal{Y}^{(t)} = \{-1, +1\}$ for classification tasks and $\mathcal{Y}^{(t)} = \mathbb{R}$ for regression tasks). Each task t has n_t training instances $\mathbf{X}^{(t)} \in \mathbb{R}^{d \times n_t}$ with corresponding labels $\mathbf{y}^{(t)} \in \mathcal{Y}^{(t)^{n_t}}$ given by $\hat{f}^{(t)}$. We assume that *a priori* the learner does not know the total number of tasks $T_{\max}$, the distribution of these tasks, or their order.

Each time step, the agent receives a batch of labeled training data for some task t, either a new task or a previously learned task. Let T denote the number of tasks encountered so far. After receiving each batch of data, the agent may be asked to make predictions on instances of any previous task. Its goal is to construct task models $f^{(1)}, \ldots, f^{(T)}$ where each $f^{(t)} : \mathbb{R}^d \mapsto \mathcal{Y}^{(t)}$ such that: (1) each $f^{(t)}$ will approximate $\hat{f}^{(t)}$ to enable the accurate prediction of labels for new instances, (2) each $f^{(t)}$ can be rapidly updated as the agent encounters additional training data for known tasks, and (3) new $f^{(t)}$'s can be added efficiently as the agent encounters new tasks. We assume that the total numbers of tasks $T_{\max}$ and data instances $\sum_{t=1}^{T_{\max}} n_t$ will be large, and so a lifelong learning algorithm must have a computational complexity to update the models that scales favorably with both quantities.

3.2. Task Structure Model for ELLA

ELLA takes a parametric approach to lifelong learning in which the prediction function $f^{(t)}(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}^{(t)})$ for each task t is governed by the task-specific parameter vector $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^d$. To model the relationships between tasks, we assume that the parameter vectors can be represented using a linear combination of shared latent model components from a knowledge repository. Many recent MTL methods employ this same technique of using a shared basis as a means to transfer knowledge between learning problems (see Section 2).

Our model of latent task structure is based on the GO-MTL model proposed by Kumar & Daumé III (2012). ELLA maintains a library of k latent model components $\mathbf{L} \in \mathbb{R}^{d \times k}$ shared between tasks. Each task parameter vector $\boldsymbol{\theta}^{(t)}$ can be represented as a linear combination of the columns of $\mathbf{L}$ according to the weight vector $\mathbf{s}^{(t)} \in \mathbb{R}^k$ (i.e., $\boldsymbol{\theta}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}$). We encourage the $\mathbf{s}^{(t)}$'s to be sparse (i.e., use few latent components) in order to ensure that each learned model component captures a maximal reusable chunk of knowledge.

Given the labeled training data for each task, we optimize the models to minimize the predictive loss over all tasks while encouraging the models to share structure. This problem is realized by the objective function:

$$e_T(\mathbf{L}) = \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \mathbf{L}\mathbf{s}^{(t)}\right), y_i^{(t)}\right) + \mu\|\mathbf{s}^{(t)}\|_1 \right\} + \lambda\|\mathbf{L}\|_{\mathrm{F}}^2 \ , \qquad (1)$$

where $(\mathbf{x}_i^{(t)}, y_i^{(t)})$ is the ith labeled training instance for task t, $\mathcal{L}$ is a known loss function, and the L_1 norm of $\mathbf{s}^{(t)}$ is used as a convex approximation to the true vector sparsity. This is similar to the model used in GO-MTL, with the modification that we average the model losses on the training data across tasks (giving rise to the $\frac{1}{T}$ term). This modification is crucial for obtaining the convergence guarantees in Section 3.6.

Since Equation 1 is not jointly convex in $\mathbf{L}$ and the $\mathbf{s}^{(t)}$'s, our goal will be to develop a procedure to arrive at a local optimum of the objective function. A common approach for computing a local optimum for objective functions of this type is to alternately perform two convex optimization steps: one in which $\mathbf{L}$ is optimized while holding the $\mathbf{s}^{(t)}$'s fixed, and another in which the $\mathbf{s}^{(t)}$'s are optimized while holding $\mathbf{L}$ fixed. These two steps are then repeated until convergence (this is the approach employed for model optimization in GO-MTL). Next, we discuss two reasons why this approach is inefficient and thus inapplicable to lifelong learning with many tasks and data instances.

The first inefficiency arises due to the explicit dependence of Equation 1 on *all* of the previous training data (through the inner summation). We remove this inefficiency by approximating Equation 1 using the second-order Taylor expansion of $\frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}\right), y_i^{(t)}\right)$ around $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$, where $\boldsymbol{\theta}^{(t)} = \arg\min_{\theta} \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}, y_i^{(t)}\right)\right)$ (that is, $\boldsymbol{\theta}^{(t)}$ is an optimal predictor learned on only the training data for task t). Plugging the second-order Taylor expansion into Equation 1 yields:

$$g_T(\mathbf{L}) = \frac{1}{T}\sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \frac{1}{n_t}\|\boldsymbol{\theta}^{(t)} - \mathbf{L}\mathbf{s}^{(t)}\|_{\mathbf{D}^{(t)}}^2 + \mu\|\mathbf{s}^{(t)}\|_1 \right\} + \lambda\|\mathbf{L}\|_{\mathrm{F}}^2 \qquad (2)$$

where

$$\mathbf{D}^{(t)} = \frac{1}{2}\nabla_{\boldsymbol{\theta},\boldsymbol{\theta}}^2 \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}\right), y_i^{(t)}\right)\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$$

$$\boldsymbol{\theta}^{(t)} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n_t}\sum_{i=1}^{n_t} \mathcal{L}\left(f\left(\mathbf{x}_i^{(t)}; \boldsymbol{\theta}\right), y_i^{(t)}\right) \ ,$$

and $\|\mathbf{v}\|_{\mathbf{A}}^2 = \mathbf{v}^\top\mathbf{A}\mathbf{v}$. In Equation 2, we have suppressed the constant term of the Taylor expansion (since it does not affect the minimum) and there is no linear term (since by construction $\boldsymbol{\theta}^{(t)}$ is a minimizer). Crucially, we have removed the dependence of the optimization on the number of data instances $n_1 \ldots n_T$ in each task. The approximation is exact in an important special case: when the model is linear and the loss function is squared loss (see Section 3.4.1).

The second inefficiency of Equation 1 is that in order to evaluate a single candidate $\mathbf{L}$, an optimization problem must be solved to recompute the value of each of the $\mathbf{s}^{(t)}$'s (which will become increasingly expensive as the number of tasks learned T increases). To overcome this problem, we modify the formulation in Equation 2 to remove the minimization over $\mathbf{s}^{(t)}$. We accomplish this by computing each of the $\mathbf{s}^{(t)}$'s when the training data for task t is last encountered, and not updating them when training on other tasks. At first glance this might seem to prevent the ability for previously learned tasks to benefit from training on later tasks (which we call *reverse transfer*); however, these tasks can benefit by subsequent modifications to $\mathbf{L}$. Later in Section 3.6, we show that this choice to update $\mathbf{s}^{(t)}$ only when we encounter training data for the respective task does not significantly affect the quality of model fit to the data as the number of tasks grows large. Using the previously computed values of $\mathbf{s}^{(t)}$ gives rise to the following optimization procedure (where we use the notation $\mathbf{L}_m$ to refer to the value of the latent components at the start of the mth iteration, and t is assumed to correspond to the particular

Algorithm 1 ELLA (k, d, λ, μ)

$T \leftarrow 0$, $\mathbf{A} \leftarrow \mathbf{zeros}_{k\times d,k\times d}$,
$\mathbf{b} \leftarrow \mathbf{zeros}_{k\times d,1}$, $\mathbf{L} \leftarrow \mathbf{zeros}_{d,k}$
while isMoreTrainingDataAvailable() **do**
  $(\mathbf{X}_{\text{new}}, \mathbf{y}_{\text{new}}, t) \leftarrow$ getNextTrainingData()
  if isNewTask(t) **then**
    $T \leftarrow T + 1$
    $\mathbf{X}^{(t)} \leftarrow \mathbf{X}_{\text{new}}$, $\mathbf{y}^{(t)} \leftarrow \mathbf{y}_{\text{new}}$
  else
    $\mathbf{A} \leftarrow \mathbf{A} - \left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)}$
    $\mathbf{b} \leftarrow \mathbf{b} - \text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right)$
    $\mathbf{X}^{(t)} \leftarrow \left[\mathbf{X}^{(t)}\ \mathbf{X}_{\text{new}}\right]$, $\mathbf{y}^{(t)} \leftarrow \left[\mathbf{y}^{(t)}; \mathbf{y}_{\text{new}}\right]$
  end if
  $\left(\boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}\right) \leftarrow$ singleTaskLearner$(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$
  $\mathbf{L} \leftarrow$ reinitializeAllZeroColumns($\mathbf{L}$)
  $\mathbf{s}^{(t)} \leftarrow$ Equation 3
  $\mathbf{A} \leftarrow \mathbf{A} + \left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)}$
  $\mathbf{b} \leftarrow \mathbf{b} + \text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right)$
  $\mathbf{L} \leftarrow \text{mat}\left(\left(\frac{1}{T}\mathbf{A} + \lambda\mathbf{I}_{k\times d,k\times d}\right)^{-1}\frac{1}{T}\mathbf{b}\right)$
end while

task for which we just received training data):

$$\mathbf{s}^{(t)} \leftarrow \arg\min_{\mathbf{s}^{(t)}} \ell(\mathbf{L}_m, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}) \quad (3)$$

$$\mathbf{L}_{m+1} \leftarrow \arg\min_{\mathbf{L}} \hat{g}_m(\mathbf{L}) \quad (4)$$

$$\hat{g}_m(\mathbf{L}) = \lambda\|\mathbf{L}\|_{\mathbf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T} \ell\left(\mathbf{L}, \mathbf{s}^{(t)}, \boldsymbol{\theta}^{(t)}, \mathbf{D}^{(t)}\right) \quad (5)$$

where

$$\ell\left(\mathbf{L}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{D}\right) = \mu\left\|\mathbf{s}\right\|_1 + \left\|\boldsymbol{\theta} - \mathbf{L}\mathbf{s}\right\|_{\mathbf{D}}^2 . \quad (6)$$

Next, we present the specific steps needed to perform the updates in the preceding equations.

3.3. Model Update for ELLA

Suppose that at the mth iteration we receive training data for task t. We must perform two steps to update our model: compute $\mathbf{s}^{(t)}$ and update $\mathbf{L}$. In order to compute $\mathbf{s}^{(t)}$, we first compute an optimal model $\boldsymbol{\theta}^{(t)}$ using only the data from task t. The details of this step will depend on the form of the model and loss function under consideration, and thus here we treat it as a black box. If the training data for a particular task arrive interleaved with other tasks and not in a single batch, it may be important to use an online single-task learning algorithm to achieve maximum scalability.

Once $\boldsymbol{\theta}^{(t)}$ has been computed, we next compute $\mathbf{D}^{(t)}$ (which is model-dependent) and re-initialize (either randomly or to one of the $\boldsymbol{\theta}^{(t)}$'s) any columns of $\mathbf{L}$ that are all-zero (which will occur if a particular latent component is currently unused). We then compute $\mathbf{s}^{(t)}$ using the current basis $\mathbf{L}_m$ by solving an L_1-regularized regression problem—an instance of the Lasso.

To update $\mathbf{L}$, we null the gradient of Equation 5 and solve for $\mathbf{L}$. This procedure yields the updated column-wise vectorization of $\mathbf{L}$ as $\mathbf{A}^{-1}\mathbf{b}$, where:

$$\mathbf{A} = \lambda\mathbf{I}_{d\times k,d\times k} + \frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{s}^{(t)}\mathbf{s}^{(t)\top}\right) \otimes \mathbf{D}^{(t)} \quad (7)$$

$$\mathbf{b} = \frac{1}{T}\sum_{t=1}^{T}\text{vec}\left(\mathbf{s}^{(t)\top} \otimes \left(\boldsymbol{\theta}^{(t)\top}\mathbf{D}^{(t)}\right)\right) . \quad (8)$$

To avoid having to sum over all tasks to compute $\mathbf{A}$ and $\mathbf{b}$ at each step, we construct $\mathbf{A}$ incrementally as new tasks arrive (see Algorithm 1 for details).

Computational Complexity: Each update begins by running a single-task learner to compute $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$; we assume that this step has complexity $O(\xi(d, n_t))$. Next, to update $\mathbf{s}^{(t)}$ requires solving an instance of the Lasso, which has complexity $O(nd\min(n, d))$, where d is the dimensionality and n is the number of data instances. Equation 3 can be seen as an instance of the Lasso in k dimensions with d data instances, for a total complexity of $O(dk^2)$. However, to formulate the Lasso problem requires computing the eigendecomposition of $\mathbf{D}^{(t)}$, which takes $O(d^3)$, and multiplying the matrix square root of $\mathbf{D}^{(t)}$ by $\mathbf{L}$, which takes $O(kd^2)$. Therefore, updating $\mathbf{s}^{(t)}$ takes time $O(d^3 + kd^2 + dk^2)$. A straightforward algorithm for updating $\mathbf{L}$ involves inverting a $(d \times k)$-by-$(d \times k)$ matrix, which has complexity $O(d^3k^3)$. However, we can exploit the fact that the updates to $\mathbf{A}$ are low-rank to derive a more efficient algorithm with complexity $O(d^3k^2)$ based on a recursive method (Yu, 1991) for updating the eigendecomposition of $\mathbf{A}$ (see Online Appendix). Therefore, using this more advanced approach, the overall complexity of each ELLA update is $O(k^2d^3 + \xi(d, n_t))$.

3.4. Base Learning Algorithms

Next, we show how two popular single-task learning algorithms can be used as the base learner for ELLA.

3.4.1. Linear Regression

In this setting $\mathbf{y}^{(t)} \in \mathbb{R}^{n_t}$, $f(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{x}$, and $\mathcal{L}$ is the squared-loss function. To apply ELLA, we compute the optimal single-task model $\boldsymbol{\theta}^{(t)}$, which is available in closed form as $\boldsymbol{\theta}^{(t)} = \left(\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}\right)^{-1}\mathbf{X}^{(t)}\mathbf{y}^{(t)}$ (assuming that $\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}$ is full-rank). $\mathbf{D}^{(t)}$ is also available in closed form as $\mathbf{D}^{(t)} = \frac{1}{2n_t}\mathbf{X}^{(t)}\mathbf{X}^{(t)\top}$. Given $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$, we simply follow Algorithm 1 to fill in the model-independent details.

3.4.2. Logistic Regression

In this setting $\mathbf{y}^{(t)} \in \{-1,+1\}^{n_t}$, $f(\mathbf{x};\boldsymbol{\theta}) = 1/\left(1+e^{-\boldsymbol{\theta}^\top \mathbf{x}}\right)$, and $\mathcal{L}$ is the log-loss function. To apply ELLA, we first use a single-task learner for logistic regression (of which there are many free and robust implementations) to compute the value of $\boldsymbol{\theta}^{(t)}$. $\mathbf{D}^{(t)}$ is then given as:

$$\mathbf{D}^{(t)} = \frac{1}{2n_t}\sum_{i=1}^{n_t}\sigma_i^{(t)}(1-\sigma_i^{(t)})\mathbf{x}_i^{(t)}\mathbf{x}_i^{(t)\top}$$
$$\sigma_i^{(t)} = \frac{1}{1+e^{-\boldsymbol{\theta}^{(t)\top}\mathbf{x}_i^{(t)}}} \ .$$

Given these formulas for $\boldsymbol{\theta}^{(t)}$ and $\mathbf{D}^{(t)}$, we follow Algorithm 1 to fill in the model-independent details.

3.5. Connection to Dictionary Learning for Sparse Coding

ELLA is closely connected to the problem of learning a dictionary online for sparse coding a set of input vectors. In fact, this problem is a special case of ELLA in which the $\boldsymbol{\theta}^{(t)}$'s are given as input instead of learned from training data and the $\mathbf{D}^{(t)}$'s are equal to the identity matrix. These simplifications yield the following objective function:

$$\beta(\mathbf{L}) = \lambda\|\mathbf{L}\|_{\mathrm{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{\theta}^{(t)} - \mathbf{L}\mathbf{s}^{(t)}\|_2^2 \qquad (9)$$
$$\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}}\left\{\mu\|\mathbf{s}\|_1 + \|\boldsymbol{\theta}^{(t)} - \mathbf{L}_t\mathbf{s}\|_2^2\right\} \ .$$

Equation 9 is identical to the equation used for efficient online dictionary learning by Mairal et al. (2009) with the one difference that we use a soft constraint on the magnitude of the entries of $\mathbf{L}$ (L_2 regularization), whereas Mairal et al. use a hard length constraint on each column of $\mathbf{L}$.

3.6. Convergence Guarantees

Here, we provide proof sketches for three results; complete proofs are available in the Online Appendix. For simplicity of exposition, our analysis is performed in the setting where ELLA receives training data for a new task at each iteration. Therefore, the number of tasks learned T is always equal to the iteration number m. Extending our analysis to the more general case outlined in Algorithm 1 is straightforward. Our convergence proof is closely modeled on the analysis by Mairal et al. (2009).

We define the expected cost of a particular $\mathbf{L}$ as

$$g(\mathbf{L}) = \mathbb{E}_{\mathbf{D}^{(t)},\boldsymbol{\theta}^{(t)}}\left[\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})\right] \ ,$$

where we use a subscript on the expectation operator to denote which part of the expression is a random variable. The expected cost represents how well a particular set of latent components can be used to represent a randomly selected task given that the knowledge repository $\mathbf{L}$ is not modified.

We show three results on the convergence of ELLA, given respectively as Propositions 1–3:

1. The latent model component matrix, $\mathbf{L}_T$, becomes increasingly stable as T increases.
2. The value of the surrogate cost function, $\hat{g}_T(\mathbf{L}_T)$, and the value of the true empirical cost function, $g_T(\mathbf{L}_T)$, converge almost surely (a.s.) as $T \to \infty$.
3. $\mathbf{L}_T$ converges asymptotically to a stationary point of the expected loss g.

These results are based on the following assumptions:

A. The tuples $\left(\mathbf{D}^{(t)},\boldsymbol{\theta}^{(t)}\right)$ are drawn *i.i.d.* from a distribution with compact support.
B. For all $\mathbf{L}$, $\mathbf{D}^{(t)}$, and $\boldsymbol{\theta}^{(t)}$, the smallest eigenvalue of $\mathbf{L}_\gamma^\top\mathbf{D}^{(t)}\mathbf{L}_\gamma$ is at least κ (with $\kappa > 0$), where γ is the set of non-zero indices of the vector $\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}}\|\boldsymbol{\theta}^{(t)} - \mathbf{L}\mathbf{s}\|^2_{\mathbf{D}^{(t)}}$. The non-zero elements of the unique minimizing $\mathbf{s}^{(t)}$ are given by: ${\mathbf{s}^{(t)}}_\gamma = \left(\mathbf{L}_\gamma^\top\mathbf{D}^{(t)}\mathbf{L}_\gamma\right)^{-1}\left(\mathbf{L}_\gamma^\top\mathbf{D}^{(t)}\boldsymbol{\theta}^{(t)} - \mu\boldsymbol{\epsilon}_\gamma\right)$, where the vector $\boldsymbol{\epsilon}_\gamma$ contains the signs of the non-zero entries of $\mathbf{s}^{(t)}$.

Proposition 1: $\mathbf{L}_{T+1} - \mathbf{L}_T = O\left(\frac{1}{T}\right)$.

Proof sketch: First, we show that the L_2 regularization term bounds the maximum magnitude of each entry of $\mathbf{L}$, and that the L_1 regularization term bounds the maximum magnitude of each entry of $\mathbf{s}^{(\mathbf{t})}$. Next, we show that $\hat{g}_T - \hat{g}_{T-1}$ is Lipschitz with constant $O\left(\frac{1}{T}\right)$. This result and the facts that $\mathbf{L}_{T-1}$ minimizes $\hat{g}_{T-1}$ and the L_2 regularization term ensures that the minimum eigenvalue of the Hessian of $\hat{g}_{T-1}$ is lower bounded by 2λ allow us to complete the proof. ■

Before stating our next proposition, we define:

$$\alpha(\mathbf{L},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}) = \arg\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}) \ , \qquad (10)$$

and introduce the following lemma:

Lemma 1:

A. $\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})$ is continuously differentiable in $\mathbf{L}$ with $\nabla_{\mathbf{L}}\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}) = -2\mathbf{D}^{(t)}\left(\boldsymbol{\theta}^{(t)} - \mathbf{L}\alpha(\mathbf{L},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})\right)\alpha(\mathbf{L},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})^\top$.
B. g is continuously differentiable with $\nabla g(\mathbf{L}) = 2\lambda\mathbf{I} + \mathbb{E}_{\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)}}\left[\nabla_{\mathbf{L}}\min_{\mathbf{s}}\ell(\mathbf{L},\mathbf{s},\boldsymbol{\theta}^{(t)},\mathbf{D}^{(t)})\right]$.
C. $\nabla_{\mathbf{L}}g(\mathbf{L})$ is Lipschitz on the space of latent model components $\mathbf{L}$.

Proof sketch: Part (A) can be easily shown using the fact that α is continuous and by applying a corollary of Theorem 4.1 as stated by Bonnans & Shapiro (1998)

London School Data The London Schools data set consists of examination scores from 15,362 students in 139 schools from the Inner London Education Authority. We treat the data from each school as a separate task. The goal is to predict the examination score of each student. We use the same feature encoding as used by Kumar & Daumé III (2012), where four school-specific categorical variables along with three student-specific categorical variables are encoded as a collection of binary features. In addition, we use the examination year and a bias term as additional features, giving each data instance $d = 27$ features.

4.2. Evaluation Procedure

Each data set was evaluated using 10 randomly generated 50/50 splits of the data between a training and hold-out test set. The particular splits were standardized across algorithms. For the online algorithms (ELLA and OMTL), we evaluated 100 randomized presentation orders of the tasks.

The parameter values of k and λ for ELLA and GO-MTL were selected independently for each algorithm and data set using a gridsearch over values of k from 1 to either 10 or $\frac{1}{4}T_{\max}$ (whichever was smaller) and values of λ from the set $\{e^{-5}, e^{-2}, e^{1}, e^{4}\}$. For ELLA, we selected the parameter values based on the training data alone. For a given training/test split, we further subdivided each training set into 10 random 50/50 sub-training/sub-validation sets and then chose parameter values that maximized the average performance on each of these 10 sub-validation sets. For GO-MTL, OMTL, and STL, the particular parameter values were selected to maximize test performance on the hold-out data averaged across all 10 random splits. Note that this procedure of choosing parameter values to maximize test performance provides ELLA with a disadvantage relative to the other algorithms.

The two parameters (burn-in time and learning rate) for OMTL were optimized using a gridsearch. The burn-in time was optimized over the set $\{50, 100, 150, \ldots, 400\}$ and the learning rate was optimized over the set $\{e^{-30}, e^{-29}, \ldots, e^{0}\}$. We report results using the LogDet update rule for OMTL, and found that the results did not vary greatly when other rules were employed; see (Saha et al., 2011) for more details on OMTL. For STL, the ridge term for either logistic or linear regression was selected by performing a gridsearch over the set $\{e^{-5}, e^{-4}, \ldots, e^{5}\}$.

Each task was presented sequentially to ELLA and OMTL, following the lifelong learning framework (Section 3.1). ELLA learned each new task from a single batch of data that contained all training instances of that task. For OMTL, which learns one instance at a time, we performed five passes over the training data for each task. We also tried using more than five passes, but the OMTL model accuracy did not increase further. GO-MTL was considered to have converged when either its objective function value decreased by less than 10^{-3} or 2,000 iterations were executed.

We measured predictive performance on the classification problems using the area under the ROC curve (AUC). This particular performance metric was chosen since both classification data sets had highly biased class distributions, and therefore other metrics like misclassification rate would only be informative for specific applications with well-specified tradeoffs between true and false positives. For regression problems, the performance was evaluated using the negative root mean-squared error (-rMSE) metric (with -rMSE, higher numbers indicate better performance). Recall that OMTL does not support regression, and so we do not evaluate it on the regression tasks.

The computational cost of each algorithm was measured using wall-clock time on a Mac Pro computer with 8GB RAM and two 6-core 2.67GHz Intel Xeon processors. We report the running time for the batch GO-MTL algorithm, and the speedup that ELLA, STL, and OMTL obtain relative to the batch algorithm both for learning *all* tasks and for learning each consecutive new task. We optimized the implementations of all algorithms to ensure a fair comparison.

4.3. Results

For classification problems, ELLA achieves nearly identical performance to GO-MTL (Table 1) while registering speedups of at least 1,350 times for learning all tasks and 38,400 times for learning each new task (Table 2). In addition, OMTL, which is specifically designed for learning efficiently online, achieved significantly worse accuracy on land mine detection and moderately worse accuracy on facial expression recognition. While OMTL did run much faster than GO-MTL, its speed did not match ELLA's. STL was the fastest approach (ELLA is necessarily slower than STL since STL is used as a subroutine inside ELLA), but had lower accuracy than ELLA.

We find similar results for the regression problems, with ELLA achieving nearly identical accuracy to GO-MTL (within 1.1% for real data and 2.3% for synthetic data; see Table 1), while achieving dramatically shorter learning times when learning all tasks (minimum speedup of 2,721 times) and each new task (minimum speedup of 378,219 times). STL was again the fastest of all approaches, but had lower accuracy.

(originally shown by Danskin (1967)). Part (B) follows directly since the tuple $(\mathbf{D}^{(t)}, \boldsymbol{\theta}^{(t)})$ is drawn from a distribution with compact support. Part (C) of the lemma crucially relies on Assumption (B), which ensures that the optimal sparse coding solution is unique. This fact, in combination with some properties that the optimal sparse coding solution must satisfy, allows us to prove that α is Lipschitz, which implies that $\nabla_{\mathbf{L}} g$ is Lipschitz due to the form of the gradient established in Parts (A) and (B). ■

Proposition 2:

A. $\hat{g}_T(\mathbf{L}_T)$ converges a.s.
B. $\hat{g}_T(\mathbf{L}_T) - g_T(\mathbf{L}_T)$ converges a.s. to 0.
C. $\hat{g}_T(\mathbf{L}_T) - g(\mathbf{L}_T)$ converges a.s. to 0.
D. $g(\mathbf{L}_T)$ converges a.s.

Proof sketch: First, we show that the sum of the positive variations of the stochastic process $u_T = \hat{g}_T(\mathbf{L}_T)$ are bounded by invoking a corollary of the Donsker theorem ((Van der Vaart, 2000) Chapter 19.2, lemma 19.36, ex. 19.7). Given this result, we apply a theorem from (Fisk, 1965) to show that u_t is a quasi-martingale that converges almost surely. The fact that u_t is a quasi-martingale along with a simple theorem of positive sequences allows us to prove part (B) of the proposition. The final two parts (C & D) can be shown due to the equivalence of g and g_T as $T \to \infty$. ■

Proposition 3: The distance between $\mathbf{L}_T$ and the set of g's stationary points converges a.s. to 0 as $T \to \infty$.

Proof sketch: We employ the fact that both the surrogate $\hat{g}_T$ and the expected cost g each have gradients that are Lipschitz with constant independent of T. This fact, in combination with the fact that $\hat{g}_T$ and g converge a.s. as $T \to \infty$, completes the proof. ■

4. Evaluation

We evaluate ELLA against three other approaches: (1) GO-MTL (Kumar & Daumé III, 2012), a batch MTL algorithm, (2) a perceptron-based approach to online multi-task learning (OMTL) (Saha et al., 2011), and (3) independent single-task learning (STL). GO-MTL provides a reasonable upper-bound on the accuracy of the models learned by ELLA (since it is a batch algorithm that optimizes all task models simultaneously). We are chiefly interested in understanding the tradeoff in accuracy between models learned with ELLA and GO-MTL, and the computational cost of learning these models. The comparison to OMTL allows us to understand how the performance of ELLA compares with another approach designed to learn efficiently in the lifelong learning setting.

4.1. Data Sets

We tested each algorithm on four multi-task data sets: (1) synthetic regression tasks, (2) land mine detection from radar images, (3) identification of three different facial movements from photographs of a subject, and (4) predicting student exam scores. Data sets (2) and (4) are benchmark data sets for MTL. We introduce data set (3) as an MTL problem for the first time.

Synthetic Regression Tasks We created a set of $T_{\max} = 100$ random tasks with $d = 13$ features and $n_t = 100$ instances per task. The task parameter vectors $\boldsymbol{\theta}^{(t)}$ were generated as a linear combination of $k = 6$ randomly generated latent components in $\mathbb{R}^{12}$. The vectors $\mathbf{s}^{(t)}$ had a sparsity level of 0.5 (i.e., half the latent components were used to construct each $\boldsymbol{\theta}^{(t)}$). The training data $\mathbf{X}^{(t)}$ was generated from a standard normal distribution. The training labels were given as $\mathbf{y}^{(t)} = {\mathbf{X}^{(t)}}^{\top} \boldsymbol{\theta}^{(t)} + \epsilon$, where each element of ϵ is independent univariate Gaussian noise. A bias term was added as the 13th feature prior to learning.

Land Mine Detection In the land mine data set (Xue et al., 2007), the goal is to detect whether or not a land mine is present in an area based on radar images. The input features are automatically extracted from radar data and consist of four-moment based features, three correlation-based features, one energy-ratio feature, one spatial variance feature, and a bias term; see (Xue et al., 2007) for more details. The data set consists of a total of 14,820 data instances divided into 29 different geographical regions. We treat each geographical region as a different task.

Facial Expression Recognition This data set is from a recent facial expression recognition challenge (Valstar et al., 2011). The goal is to detect the presence or absence of three different facial action units (#5: upper lid raiser, #10: upper lip raiser, and #12: lip corner pull) from an image of a subject's face. We chose this combination of action units to be a challenge, since two of the action units involve the lower face, suggesting a high potential for transfer, while the other is an upper face action unit, suggesting a low potential for transfer. Each task involves recognizing one of the three action units for one of seven subjects, yielding a total of 21 tasks, each with 450–999 images. To represent the images, we utilized a Gabor pyramid with a frequency bandwidth of 0.7 octaves, orientation bandwidth of 120 degrees, four orientations, 576 locations, and two spatial scales, yielding a total of 2,880 Gabor features for each image. We reduced the raw Gabor outputs to 100 dimensions using PCA, and added a bias term to produce the input features.

Table 1. The accuracy of ELLA, OMTL, and STL relative to batch multi-task learning (GO-MTL), showing that ELLA achieves nearly equal accuracy to batch MTL and better accuracy than OMTL. The N/A's indicate that OMTL does not handle regression problems. The standard deviation of a value is given after the ± symbol.

Dataset	Problem Type	Batch MTL Accuracy	ELLA Relative Accuracy	OMTL Relative Accuracy	STL Relative Accuracy
Land Mine	Classification	0.7802 ± 0.013 (AUC)	$99.73 \pm 0.7\%$	$82.2 \pm 3.0\%$	$97.97 \pm 1.5\%$
Facial Expr.	Classification	0.6577 ± 0.021 (AUC)	$99.37 \pm 3.1\%$	$97.58 \pm 3.8\%$	$97.34 \pm 3.9\%$
Syn. Data	Regression	-1.084 ± 0.006 (-rMSE)	$97.74 \pm 2.7\%$	N/A	$92.91 \pm 1.5\%$
London Sch.	Regression	-10.10 ± 0.066 (-rMSE)	$98.90 \pm 1.5\%$	N/A	$97.20 \pm 0.4\%$

Table 2. The running time of ELLA, OMTL, and STL as compared to batch multi-task learning (GO-MTL), showing that ELLA achieves three orders of magnitude speedup in learning all tasks, and four-to-five orders of magnitude speedup in learning each consecutive new task. The N/A's indicate that OMTL does not handle regression. Speedup was measured relative to the batch method using optimized implementations. The standard deviation of a value is given after the ±.

Dataset	Batch Runtime (seconds)	ELLA All Tasks (speedup)	ELLA New Task (speedup)	OMTL All Tasks (speedup)	OMTL New Task (speedup)	STL All Tasks (speedup)	STL New Task (speedup)
Land Mine	231±6.2	1,350±58	39,150±1,682	22±0.88	638±25	3,342±409	96,918±11,861
Facial Expr.	2,200±92	1,828±100	38,400±2,100	948±65	19,900±1,360	8,511±1,107	178,719±23,239
Syn. Data	1,300±141	5,026±685	502,600±68,500	N/A	N/A	156,489±17,564	1.6E6±1.8E5
London Sch.	715±36	2,721±225	378,219±31,275	N/A	N/A	36,000±4,800	5.0E6±6.7E5

Recall that ELLA does not re-optimize the value of $\mathbf{s}^{(t)}$ unless it receives new training data for task t. Therefore, in each experiment, the value of $\mathbf{s}^{(t)}$ is set when the training data for that task is presented and never readjusted. Although the values of $\mathbf{s}^{(t)}$ are not updated, it is still possible that previously learned task models can benefit from training on subsequent tasks through modifications to $\mathbf{L}$.

To assess whether this phenomenon of reverse transfer occurred, we computed the change in accuracy from when a task was first learned until after *all* tasks had been learned. A positive change in accuracy for a task indicates that reverse transfer did occur. Figure 2 shows this change in accuracy as a function of position in the task sequence, revealing that reverse transfer occurred reliably in all data sets and that reverse transfer is most beneficial for tasks that were learned early (when the total amount of training data seen was low). Most importantly, these results show that, with few exceptions, subsequent learning did not reduce the performance of models that were learned early.

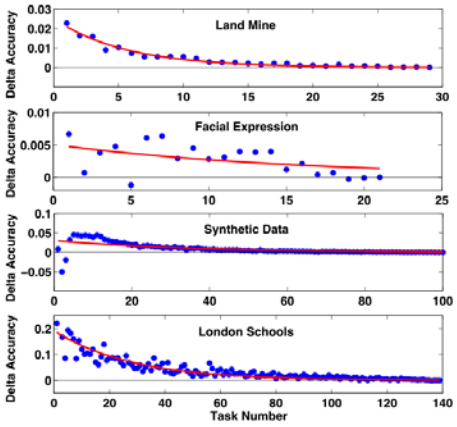


Figure 2. The change in accuracy from when a task is first learned until all tasks have been learned, as a function of position in the task sequence. Plotted lines are the best fitting exponential curve.

5. Conclusion

We have presented an efficient algorithm for lifelong learning (ELLA) that provides nearly identical accuracy to batch MTL, while requiring three orders of magnitude less runtime. Also, ELLA is more flexible, faster, and achieves better accuracy than a competing method for online MTL. We have shown that ELLA works well on synthetic data as well as three multi-task problems. Additionally, we discussed ELLA's connections to online dictionary learning for sparse coding, and presented theoretical guarantees that illuminate the reasons for ELLA's strong performance. Our future work will include extending ELLA to settings besides linear and logistic models and automatically adjusting the basis size k as it learns more tasks.

Acknowledgements

This research was supported by ONR grant N00014-11-1-0139. We thank Terran Lane, Diane Oyen, and the anonymous reviewers for their helpful feedback.

References

Bonnans, J.F. and Shapiro, A. Optimization problems with perturbations: A guided tour. *SIAM review*, 40 (2):228–264, 1998.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., and Mitchell., T.M. Toward an architecture for never-ending language learning. In *Proceedings of the 24th Conference on Artificial Intelligence*. AAAI Press, 2010.

Danskin, J.M. *The theory of max-min and its application to weapons allocation problems*, volume 5 of Econometrics and Operations Research. Springer-Verlag, 1967.

Fisk, D.L. Quasi-martingales. *Transactions of the American Mathematical Society*, 120(3):369–389, 1965.

Kumar, A. and Daumé III, H. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1383–1390. Omnipress, 2012.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 689–696. ACM, 2009.

Rai, P. and Daumé III, H. Infinite predictor subspace models for multitask learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 613–620, 2010.

Ring, M.B. CHILD: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

Saha, A., Rai, P., Daumé III, H., and Venkatasubramanian, S. Online learning of multiple tasks and their relationships. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 643–651, 2011.

Simm, J., Sugiyama, M., and Kato, T. Computationally efficient multi-task learning with least-squares probabilistic classifiers. *IPSJ Transactions on Computer Vision and Applications*, 3:1–8, 2011.

Sutton, R., Koop, A., and Silver, D. On the role of tracking in stationary environments. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 871–878, Corvallis, OR, 2007. ACM.

Thrun, S. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996.

Thrun, S. and O'Sullivan, J. Discovering structure in multiple learning tasks: the TC algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pp. 489–497. Morgan Kaufmann, 1996.

Valstar, M.F., Jiang, B., Méhu, M., Pantic, M., and Scherer, K. The first facial expression recognition and analysis challenge. In *Proceedings of the IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 921–926. IEEE, 2011.

Van der Vaart, A.W. *Asymptotic statistics*, volume 3. Cambridge University Press, 2000.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

Yu, K.B. Recursive updating the eigenvalue decomposition of a covariance matrix. *IEEE Transactions on Signal Processing*, 39(5):1136–1145, 1991.

Zhang, J., Ghahramani, Z., and Yang, Y. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

Appendix B: Bou Aamar et al., 2015a

Unsupervised Cross-Domain Transfer in Policy Gradient Reinforcement Learning via Manifold Alignment

Haitham Bou Ammar
Univ. of Pennsylvania
haithamb@seas.upenn.edu

Eric Eaton
Univ. of Pennsylvania
eeaton@cis.upenn.edu

Paul Ruvolo
Olin College of Engineering
paul.ruvolo@olin.edu

Matthew E. Taylor
Washington State Univ.
taylorm@eecs.wsu.edu

Abstract

The success of applying policy gradient reinforcement learning (RL) to difficult control tasks hinges crucially on the ability to determine a sensible initialization for the policy. Transfer learning methods tackle this problem by reusing knowledge gleaned from solving other related tasks. In the case of multiple task domains, these algorithms require an *inter-task mapping* to facilitate knowledge transfer across domains. However, there are currently no general methods to learn an inter-task mapping without requiring either background knowledge that is not typically present in RL settings, or an expensive analysis of an exponential number of inter-task mappings in the size of the state and action spaces.

This paper introduces an autonomous framework that uses unsupervised manifold alignment to learn inter-task mappings and effectively transfer samples between different task domains. Empirical results on diverse dynamical systems, including an application to quadrotor control, demonstrate its effectiveness for cross-domain transfer in the context of policy gradient RL.

Introduction

Policy gradient reinforcement learning (RL) algorithms have been applied with considerable success to solve high-dimensional control problems, such as those arising in robotic control and coordination (Peters & Schaal 2008). These algorithms use gradient ascent to tune the parameters of a policy to maximize its expected performance. Unfortunately, this gradient ascent procedure is prone to becoming trapped in local maxima, and thus it has been widely recognized that initializing the policy in a sensible manner is crucial for achieving optimal performance. For instance, one typical strategy is to initialize the policy using human demonstrations (Peters & Schaal 2006), which may be infeasible when the task cannot be easily solved by a human. This paper explores a different approach: instead of initializing the policy at random (i.e., *tabula rasa*) or via human demonstrations, we instead use transfer learning (TL) to initialize the policy for a new target domain based on knowledge from one or more source tasks.

In RL transfer, the source and target tasks may differ in their formulations (Taylor & Stone 2009). In particular, when the source and target tasks have different state and/or action spaces, an *inter-task mapping* (Taylor et al. 2007a) that describes the relationship between the two tasks is typically needed. This paper introduces a framework for autonomously learning an inter-task mapping for cross-domain transfer in policy gradient RL. First, we learn an inter-state mapping (i.e., a mapping between states in two tasks) using unsupervised manifold alignment. Manifold alignment provides a powerful and general framework that can discover a shared latent representation to capture intrinsic relations between different tasks, irrespective of their dimensionality. The alignment also yields an implicit inter-action mapping that is generated by mapping *tracking* states from the source to the target. Given the mapping between task domains, source task trajectories are then used to initialize a policy in the target task, significantly improving the speed of subsequent learning over an uninformed initialization.

This paper provides the following contributions. First, we introduce a novel unsupervised method for learning inter-state mappings using manifold alignment. Second, we show that the discovered subspace can be used to initialize the target policy. Third, our empirical validation conducted on four dissimilar and dynamically chaotic task domains (e.g., controlling a three-link cart-pole and a quadrotor aerial vehicle) shows that our approach can *a*) automatically learn an inter-state mapping across MDPs from the same domain, *b*) automatically learn an inter-state mapping across MDPs from *very different* domains, and *c*) transfer informative initial policies to achieve higher initial performance and reduce the time needed for convergence to near-optimal behavior.

Related Work

Learning an inter-task mapping has been of major interest in the transfer learning community because of its promise of autonomous transfer between very different tasks (Taylor & Stone 2009). However, the majority of existing work assumes that *a*) the source task and target task are similar enough that no mapping is needed (Banerjee & Stone 2007; Konidaris & Barto 2007), or *b*) an inter-task mapping is provided to the agent (Taylor et al. 2007a; Torrey et al. 2008). The main difference between these methods and this paper is that we are interested in *learning* a mapping between tasks.

There has been some recent work on learning such mappings. For example, mappings may be based on seman-

tic knowledge about state features between two tasks (Liu & Stone 2006), background knowledge about the range or type of state variables (Taylor et al. 2007b), or transition models for each possible mapping could be generated and tested (Taylor et al. 2008). However, there are currently no general methods to learn an inter-task mapping without requiring either background knowledge that is not typically present in RL settings, or an expensive analysis of an exponential number (in the size of the action and state variable sets) of inter-task mappings. We overcome these issues by automatically discovering high-level features and using them to transfer knowledge between agents without suffering from an exponential explosion.

In previous work, we used sparse coding, sparse projection, and sparse Gaussian processes to learn an inter-task mapping between MDPs with arbitrary variations (Bou Ammar et al. 2012). However, this previous work relied on a Euclidean distance correlation between source and target task triplets, which may fail for highly dissimilar tasks. Additionally, it placed restrictions on the inter-task mapping that reduced the flexibility of the learned mapping. In other related work, Bósci *et al.* (2013) use manifold alignment to assist in transfer. The primary differences with our work are that the authors *a*) focus on transferring models between different robots, rather than policies/samples, and *b*) rely on source and target robots that are qualitatively similar.

Background

Reinforcement Learning problems involve an agent choosing sequential actions to maximize its expected return. Such problems are typically formalized as a Markov decision process (MDP) $\mathcal{T} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_0, \mathcal{P}, r \rangle$, where $\mathcal{S}$ is the (potentially infinite) set of states, $\mathcal{A}$ is the set of actions that the agent may execute, $\mathcal{P}_0 : \mathcal{S} \rightarrow [0, 1]$ is a probability distribution over the initial state, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition probability function describing the task dynamics, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function measuring the performance of the agent. A policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is defined as a conditional probability distribution over actions given the current state. The agent's goal is to find a policy $\pi^\star$ which maximizes the average expected reward:

$$\pi^\star = \arg\max_\pi \ \mathrm{E}\left[\frac{1}{H} \sum_{t=1}^{H} r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \middle| \pi \right] \qquad (1)$$

$$= \arg\max_\pi \int_{\mathbb{T}} p_\pi(\boldsymbol{\tau}) \mathcal{R}(\boldsymbol{\tau}) d\boldsymbol{\tau} \ ,$$

where $\mathbb{T}$ is the set of all possible trajectories with horizon H,

$$\mathcal{R}(\boldsymbol{\tau}) = \frac{1}{H} \sum_{t=1}^{H} r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \ \text{, and} \qquad (2)$$

$$p_\pi(\boldsymbol{\tau}) = \mathcal{P}_0(\boldsymbol{s}_1) \prod_{t=1}^{H} \mathcal{P}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t) \ \pi(\boldsymbol{a}_t | \boldsymbol{s}_t) \ . \qquad (3)$$

Policy Gradient methods (Sutton et al. 1999; Peters et al. 2005) represent the agent's policy π as a function defined over a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ of control parameters and a vector of state features given by the transformation $\boldsymbol{\Phi} : \mathcal{S} \rightarrow \mathbb{R}^m$. By substituting this parameterization of the control policy into Eqn. (2), we can compute the parameters of the optimal policy as $\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$, where $\mathcal{J}(\boldsymbol{\theta}) = \int_{\mathbb{T}} p_{\pi(\boldsymbol{\theta})}(\boldsymbol{\tau}) \mathcal{R}(\boldsymbol{\tau}) d\boldsymbol{\tau}$. To maximize $\mathcal{J}$, many policy gradient methods employ standard supervised function approximation to learn $\boldsymbol{\theta}$ by following an estimated gradient of a lower bound on the expected return of $\mathcal{J}(\boldsymbol{\theta})$.

Policy gradient algorithms have gained attention in the RL community in part due to their successful applications on real-world robotics (Peters et al. 2005). While such algorithms have a low computational cost per update, high-dimensional problems require many updates (by acquiring new rollouts) to achieve good performance. Transfer learning can reduce this data requirement and accelerate learning.

Since policy gradient methods are prone to becoming stuck in local maxima, it is crucial that the policy be initialized in a sensible fashion. A common technique (Peters & Schaal 2006; Argall et al. 2009) for policy initialization is to first collect demonstrations from a human controlling the system, then use supervised learning to fit policy parameters that maximize the likelihood of the human-demonstrated actions, and finally use the fitted parameters as the initial policy parameters for a policy gradient algorithm. While this approach works well in some settings, it is inapplicable in several common cases: *a*) when it is difficult to instrument the system in question so that a human can successfully perform a demonstration, *b*) when an agent is constantly faced with new tasks, making gathering human demonstrations for each new task impractical, or *c*) when the tasks in question cannot be intuitively solved by a human demonstrator.

The next section introduces a method for using transfer learning to initialize the parameters of a policy in a way that is not susceptible to these limitations. Our experimental results show that this method of policy initialization, when compared to random policy initialization, is able to not only achieve better initial performance, but also obtain a higher performing policy when run until convergence.

Policy Gradient Transfer Learning

Transfer learning aims to improve learning times and/or behavior of an agent on a new target task $\mathcal{T}^{(T)}$ by reusing knowledge from a solved source task $\mathcal{T}^{(S)}$. In RL settings, each task is described by an MDP: task $\mathcal{T}^{(S)} = \langle \mathcal{S}^{(S)}, \mathcal{A}^{(S)}, \mathcal{P}_0^{(S)}, \mathcal{P}^{(S)}, r^{(S)} \rangle$ and $\mathcal{T}^{(T)} = \langle \mathcal{S}^{(T)}, \mathcal{A}^{(T)}, \mathcal{P}_0^{(T)}, \mathcal{P}^{(T)}, r^{(T)} \rangle$. One way in which knowledge from a solved source task can be leveraged to solve the target task is by mapping optimal ⟨state, action, next state⟩ triples from the source task into the state and action spaces of the target task. Transferring optimal triples in this way allows us to both provide a better jumpstart and learning ability to the target agent, based on the source agent's ability.

While the preceding idea is attractive, complexities arise when the source and target tasks have different state and/or action spaces. In this case, one must define an inter-task mapping $\boldsymbol{\chi}$ in order to translate optimal triples from the source to the target task. Typically (Taylor & Stone 2009), $\boldsymbol{\chi}$ is defined by two sub-mappings: (1) an inter-state mapping $\boldsymbol{\chi}_{\mathcal{S}}$ and (2) an inter-action mapping $\boldsymbol{\chi}_{\mathcal{A}}$.

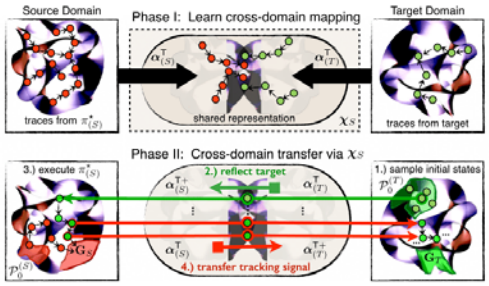


Figure 1: Transfer is split into two phases: (I) learning the inter-state mapping χ_S via manifold alignment, and (II) initializing the target policy via mapping the source task policy.

By adopting an RL framework where policies are state-feedback controllers, we show that we can use optimal state trajectories from the source task to intelligently initialize a control policy in the target task, without needing to explicitly construct an inter-action mapping. We accomplish this by learning a (pseudo-invertible) inter-state mapping between the state spaces of a pair of tasks using manifold alignment, which can then be used to transfer optimal sequences of states to the target. The fact that our algorithm does not require learning an explicit inter-action mapping significantly reduces its computational complexity.

Our approach consists of two phases (Figure 1). First, using traces gathered in the source and target tasks, we learn an inter-state mapping $\boldsymbol{\chi}_{\mathcal{S}}$ using manifold alignment ("Phase I" in Figure 1). To perform this step, we adapt the Unsupervised Manifold Alignment (UMA) algorithm (Wang & Mahadevan 2009), as detailed in the next section. Second, we use $\boldsymbol{\chi}_{\mathcal{S}}$ to project state trajectories from the source to the target task ("Phase II" in Figure 1). These projected state trajectories define a set of a tracking trajectories for the target task that allow us to perform one step of policy gradient improvement in the target task. This policy improvement step intelligently initializes the target policy, which results in superior learning performance than starting from a randomly initialized policy, as shown in our experiments. Although we focus on policy gradient methods, our approach could easily be adapted to other policy search methods (e.g., PoWER, REPS, etc.; see Kober et al. 2013).

Learning an Inter-State Mapping

Unsupervised Manifold Alignment (UMA) is a technique that efficiently discovers an alignment between two datasets (Wang & Mahadevan 2009). UMA was developed to align datasets for knowledge transfer between two supervised learning tasks. Here, we adapt UMA to an RL setting by aligning source and target task state spaces with potentially different dimensions m_S and m_T. To learn $\boldsymbol{\chi}_{\mathcal{S}}$ relating $\mathcal{S}^{(S)}$ and $\mathcal{S}^{(T)}$, trajectories of states in the source task, $\boldsymbol{\tau}^{\star}_{(S)} = \left\{\mathbf{s}_1^{(i),(S)\star}, \ldots, \mathbf{s}_{H_S}^{(i),(S)\star}\right\}_{i=1}^{n_S}$, are obtained by following $\pi^{\star}_{(S)}$, and trajectories of states in the target task, $\boldsymbol{\tau}_{(T)} = \left\{s_1^{(j),(T)}, \ldots, s_{H_T}^{(j),(T)}\right\}_{j=1}^{n_T}$, are obtained by utilizing $\pi_{(T)}$, which is initialized using randomly selected policy parameters. For simplicity of exposition, we assume that trajectories in the source domain have length H_S and those in the target domain have length H_T; however, our algorithm is capable of handling variable-length trajectories. We are interested in the setting where data is scarcer in the target task than in the source task (i.e., $n_T \ll n_S$).

Given trajectories from both the source and target tasks, we flatten the trajectories (i.e., we treat the states as unordered) and then apply the task-specific state transformation to obtain two sets of state feature vectors, one for the source task and one for the target task. Specifically, we create the following sets of points:

$$\mathbf{X}^{(S)} = \left\{\Phi^{(S)}\left(\mathbf{s}_1^{(1)(S)\star}\right), \ldots, \Phi^{(S)}\left(\mathbf{s}_{H_S}^{(1)(S)\star}\right), \Phi^{(S)}\left(\mathbf{s}_1^{(n_S)(S)\star}\right), \ldots, \Phi^{(S)}\left(\mathbf{s}_{H_S}^{(n_S)(S)\star}\right)\right\}$$

$$\mathbf{X}^{(T)} = \left\{\Phi^{(T)}\left(\mathbf{s}_1^{(1)(T)}\right), \ldots, \Phi^{(T)}\left(\mathbf{s}_{H_T}^{(1)(T)}\right), \Phi^{(T)}\left(\mathbf{s}_1^{(n_T)(T)}\right), \ldots, \Phi^{(T)}\left(\mathbf{s}_{H_T}^{(n_T)(T)}\right)\right\} .$$

Given $\mathbf{X}^{(S)} \in \mathbb{R}^{m_S \times (H_S \times n_S)}$, $\mathbf{X}^{(T)} \in \mathbb{R}^{m_T \times (H_T \times n_T)}$, we can apply the UMA algorithm (Wang & Mahadevan 2009) with minimal modification, as described next.

Unsupervised Manifold Alignment (UMA) The first step of applying UMA to learn the inter-state mapping is to represent each transformed state in both the source and target tasks in terms of its local geometry. We use the notation $\mathbf{R}_{\mathbf{x}_i^{(S)}} \in \mathbb{R}^{(k+1)\times(k+1)}$ to refer to the matrix of pairwise Euclidean distances among the k-nearest neighbors of $\mathbf{x}_i^{(S)} \in \mathbf{X}^{(S)}$. Similarly, $\mathbf{R}_{\mathbf{x}_j^{(T)}}$ refers to the equivalent matrix of distances for the k-nearest neighbors of $\mathbf{x}_j^{(T)} \in \mathbf{X}^{(T)}$.

The relations between local geometries in $\mathbf{X}^{(S)}$ and $\mathbf{X}^{(T)}$ are represented by the matrix $\mathbf{W} \in \mathbb{R}^{(n_S \times H_S)\times(n_T \times H_T)}$ with $w_{i,j} = \exp\left\{-\text{dist}\left(\mathbf{R}_{\mathbf{x}_i^{(S)}}, \mathbf{R}_{\mathbf{x}_j^{(T)}}\right)\right\}$ and distance metric

$$\text{dist}\left(\mathbf{R}_{\mathbf{x}_i^{(S)}}, \mathbf{R}_{\mathbf{x}_j^{(T)}}\right) = \min_{1\le h\le k!}\left[\min\left(\left\|\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h - \gamma_1\mathbf{R}_{\mathbf{x}_i^{(S)}}\right\|_{\text{F}}, \left\|\mathbf{R}_{\mathbf{x}_i^{(S)}} - \gamma_2\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h\right\|_{\text{F}}\right)\right] . \quad (4)$$

We use the notation $\wr\cdot\wr_h$ to denote the h^{th} variant of the $k!$ permutations of the rows and columns of the input matrix, $||\cdot||_{\text{F}}$ is the Frobenius norm, and γ_1 and γ_2 are defined as:

$$\gamma_1 = \frac{\text{tr}\left(\mathbf{R}^{\mathsf{T}}_{\mathbf{x}_i^{(S)}}\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h\right)}{\text{tr}\left(\mathbf{R}^{\mathsf{T}}_{\mathbf{x}_i^{(S)}}\mathbf{R}_{\mathbf{x}_i^{(S)}}\right)} \qquad \gamma_2 = \frac{\text{tr}\left(\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h^{\mathsf{T}}\mathbf{R}_{\mathbf{x}_i^{(S)}}\right)}{\text{tr}\left(\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h^{\mathsf{T}}\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h\right)} .$$

To align the manifolds, UMA computes the joint Laplacian

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{\mathbf{X}^{(S)}} + \mu\mathbf{\Gamma}^{(1)} & -\mu\mathbf{\Gamma}^{(2)} \\ -\mu\mathbf{\Gamma}^{(3)} & \mathbf{L}_{\mathbf{X}^{(T)}} + \mu\mathbf{\Gamma}^{(4)} \end{pmatrix} \quad (5)$$

with diagonal matrices $\mathbf{\Gamma}^{(1)} \in \mathbb{R}^{(n_S \times H_S)\times(n_S \times H_S)}$ and $\mathbf{\Gamma}^{(4)} \in \mathbb{R}^{(n_T \times H_T)\times(n_T \times H_T)}$, where $\mathbf{\Gamma}^{(1)}_{i,i} = \sum_j w_{i,j}$ and $\mathbf{\Gamma}^{(4)}_{j,j} = \sum_i w_{i,j}$. The matrices $\mathbf{\Gamma}^{(2)} \in \mathbb{R}^{(n_S \times H_S)\times(n_T \times H_T)}$ and $\mathbf{\Gamma}^{(3)} \in \mathbb{R}^{(n_T \times H_T)\times(n_S \times H_S)}$ join the two manifolds with $\mathbf{\Gamma}^{(2)}_{i,j} = w_{i,j}$ and $\mathbf{\Gamma}^{(3)}_{i,j} = w_{j,i}$.

Additionally, the non-normalized Laplacians $\mathbf{L}_{\mathbf{X}^{(S)}}$ and $\mathbf{L}_{\mathbf{X}^{(T)}}$ are defined as: $\mathbf{L}_{\mathbf{X}^{(S)}} = \mathbf{D}_{\mathbf{X}^{(S)}} - \mathbf{W}_{\mathbf{X}^{(S)}}$ and $\mathbf{L}_{\mathbf{X}^{(T)}} = \mathbf{D}_{\mathbf{X}^{(T)}} - \mathbf{W}_{\mathbf{X}^{(T)}}$, where $\mathbf{D}_{\mathbf{X}^{(S)}} \in \mathbb{R}^{(n_S \times H_S)\times(n_S \times H_S)}$ is a diagonal matrix with $\mathbf{D}^{(i,i)}_{\mathbf{X}^{(S)}} = \sum_j w^{(S)}_{i,j}$ and, similarly, $\mathbf{D}^{(i,i)}_{\mathbf{X}^{(T)}} = \sum_j w^{(T)}_{i,j}$. The matrices $\mathbf{W}^{(S)}$ and $\mathbf{W}^{(T)}$ represent the similarity in the source and target task state spaces respectively and can be computed similar to $\mathbf{W}$.

To join the manifolds, UMA first defines two matrices:

$$\mathbf{Z} = \begin{pmatrix} \boldsymbol{\tau}^{\star}_{(S)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\tau}_{(T)} \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} \mathbf{D}_{\mathcal{S}^{(S)}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\mathcal{S}^{(T)}} \end{pmatrix} . \quad (6)$$

Given $\mathbf{Z}$ and $\mathbf{D}$, UMA computes optimal projections to reduce the dimensionality of the joint structure by taking the d minimum eigenvectors $\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_d$ of the generalized eigenvalue decomposition $\mathbf{ZLZ}^T\boldsymbol{\zeta} = \lambda\mathbf{ZDZ}^T\boldsymbol{\zeta}$. The optimal projections $\boldsymbol{\alpha}_{(S)}$ and $\boldsymbol{\alpha}_{(T)}$ are then given as the first d_1 and d_2 rows of $[\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_d]$, respectively.

Given the embedding discovered by UMA, we can then define the inter-state mapping as:

$$\boldsymbol{\chi}_{\mathcal{S}}[\cdot] = \boldsymbol{\alpha}^{\mathsf{T}+}_{(T)}\boldsymbol{\alpha}^{\mathsf{T}}_{(S)}[\cdot] \ . \quad (7)$$

The inverse of the inter-state mapping (to project target states to the source task) can be determined by taking the pseudo-inverse of Eqn. (7), yielding $\boldsymbol{\chi}^{+}_{\mathcal{S}}[\cdot] = \boldsymbol{\alpha}^{\mathsf{T}+}_{(S)}\boldsymbol{\alpha}^{\mathsf{T}}_{(T)}[\cdot]$.

Intuitively, this approach aligns the important regions of the source task's state space (sampled based on optimal source trajectories) with the state space explored so far in the target task. Although actions were ignored in constructing the manifolds, the aligned representation implicitly captures local state transition dynamics within each task (since the states came from trajectories), providing a mechanism to transfer trajectories between tasks, as we describe next.

Policy Transfer and Improvement

Next, we discuss the procedure for initializing the target policy, $\pi_{(T)}$. We consider a model-free setting in which the policy is linear over a set of (potentially) non-linear state feature functions modulated by Gaussian noise (where the magnitude of the noise balances exploration and exploitation). Specifically, we can write the source and target policies as:

$$\pi^{\star}_{(S)}\left(\boldsymbol{s}^{(S)}_t\right) = \mathbf{\Phi}^{(S)}\left(\boldsymbol{s}^{(S)}_t\right)^{\mathsf{T}}\boldsymbol{\theta}^{(S)\star} + \boldsymbol{\epsilon}^{(S)}_t$$

$$\pi_{(T)}\left(\boldsymbol{s}^{(T)}_t\right) = \mathbf{\Phi}^{(T)}\left(\boldsymbol{s}^{(T)}_t\right)^{\mathsf{T}}\boldsymbol{\theta}^{(T)} + \boldsymbol{\epsilon}^{(T)}_t \ ,$$

where $\boldsymbol{\epsilon}^{(S)}_t \sim \mathcal{N}\left(0, \Sigma^{(S)}\right)$ and $\boldsymbol{\epsilon}^{(T)}_t \sim \mathcal{N}\left(0, \Sigma^{(T)}\right)$.

To initialize $\pi_{(T)}$, we first sample m initial target trajectories $\mathcal{D}^{(T)} = \left\{\boldsymbol{\tau}^{(T)}_i\right\}_{i=1}^{m}$ from the target task using a randomly initialized policy (these can be newly sampled trajectories or simply the ones used to do the initial manifold alignment step). Next, we map the set of initial states in $\mathcal{D}^{(T)}$ to the source task using $\boldsymbol{\chi}^{+}_{\mathcal{S}}$. We then run the optimal source policy starting from each of these mapped initial states to produce a set of m optimal state trajectories in the source task. Finally, the resulting state trajectories are mapped back to the target task using $\boldsymbol{\chi}_{\mathcal{S}}$ to generate a set of reflected state-trajectories in the target task, $\tilde{\mathcal{D}}^{(T)} = \left\{\tilde{\boldsymbol{\tau}}^{(T)}_i\right\}_{i=1}^{m}$. For clarity, we assume that all trajectories are of length H; however, this is not a fundamental limitation of our algorithm.

We define the following transfer cost function:

$$\mathcal{J}_{\mathcal{T}^{(S)}\rightarrow\mathcal{T}^{(T)}}\left(\boldsymbol{\theta}^{(T)}\right) = \sum_{i=1}^{m} p_{\boldsymbol{\theta}^{(T)}}\left(\boldsymbol{\tau}^{(T)}_i\right)\hat{\mathcal{R}}_{(T)}\left(\boldsymbol{\tau}^{(T)}_i, \tilde{\boldsymbol{\tau}}^{(T)}_i\right) \quad (8)$$

where $\hat{\mathcal{R}}_{(T)}$ is a cost function that penalizes deviations between the initial sampled trajectories in the target task and the reflected optimal trajectories:

$$\hat{\mathcal{R}}_{(T)}\left(\boldsymbol{\tau}^{(T)}, \tilde{\boldsymbol{\tau}}^{(T)}\right) = \frac{1}{H}\sum_{t=1}^{H}\left|\left|\boldsymbol{s}^{(T)}_t - \tilde{\boldsymbol{s}}^{(T)}_t\right|\right|^2_2 \ . \quad (9)$$

Minimizing, Eqn. (8) is equivalent to attaining a target policy parameterization $\boldsymbol{\theta}^{(T)}$ such that $\pi_{(T)}$ follows the reflected trajectories $\tilde{\mathcal{D}}^{(T)}$. Further, Eqn. (8) is in exactly the form required to apply standard off-the-shelf policy gradient algorithms to minimize the transfer cost. The Manifold Alignment Cross-Domain Transfer for Policy Gradients (MAXDT-PG) framework is detailed[1] in Algorithm 1.

Special Cases

Our work can be seen as an extension of the simpler model-based case with a linear-quadratic regulator (LQR) (Bemporad et al. 2002) policy, which is derived and explained in the online appendix[2] accompanying this paper. Although the assumptions made by the model-based case seem restrictive, the analysis in the appendix covers a wide range of applications. These, for example, include: *a*) the case in which a dynamical model is provided beforehand, or *b*) the case in which model-based RL algorithms are adopted (see Buşoniu et al. 2010). In the main paper, however, we consider the more general model-free case.

Experiments and Results

To assess MAXDT-PG's performance, we conduced experiments on transfer both between tasks in the same domain as well as between tasks in different domains. Also, we studied

[1]Lines 9-11 of Algorithm 1 require interaction with the target domain (or a simulator) for acquiring the optimal policy. Such an assumption is common to policy gradient methods, where at each iteration, data is gathered and used to iteratively improve the policy.

[2]The online appendix is available on the authors' websites.

Algorithm 1 Manifold Alignment Cross-Domain Transfer for Policy Gradients (MAXDT-PG)

Inputs: Source and target tasks $\mathcal{T}^{(S)}$ and $\mathcal{T}^{(T)}$, optimal source policy $\pi^{\star}_{(S)}$, # source and target traces n_S and n_T, # nearest neighbors k, # target rollouts z_T, initial # of target states m.

Learn $\chi_{\mathcal{S}}$:

1: Sample n_S optimal source traces, $\boldsymbol{\tau}^{\star}_{(S)}$, and n_T random target traces, $\boldsymbol{\tau}_{(T)}$

2: Using the modified UMA approach, learn $\boldsymbol{\alpha}_{(S)}$ and $\boldsymbol{\alpha}_{(T)}$ to produce $\chi_{\mathcal{S}} = \boldsymbol{\alpha}^{\mathsf{T}+}_{(T)}\boldsymbol{\alpha}^{\mathsf{T}}_{(S)}[\cdot]$

Transfer & Initialize Policy:

3: Collect m initial target states $\boldsymbol{s}^{(T)}_1 \sim \mathcal{P}^{(T)}_0$

4: Project these m states to the source by applying $\chi^{+}_{\mathcal{S}}[\cdot]$

5: Apply the optimal source policy $\pi^{\star}_{(S)}$ on these projected states to collect $\mathcal{D}^{(S)} = \left\{\boldsymbol{\tau}^{(S)}_{(i)}\right\}^{m}_{i=1}$

6: Project the samples in $\mathcal{D}^{(S)}$ to the target using $\chi_{\mathcal{S}}[\cdot]$ to produce tracking target traces $\tilde{\mathcal{D}}^{(T)}$

7: Compute tracking rewards using Eqn. (9)

8: Use policy gradients to minimize Eqn. (8), yielding $\boldsymbol{\theta}^{(0)}_{(T)}$

Improve Policy:

9: Start with $\boldsymbol{\theta}^{(0)}_{(T)}$ and sample z_T target rollouts

10: Follow policy gradients (e.g., episodic REINFORCE) but using target rewards $\mathcal{R}^{(T)}$

11: Return optimal target policy parameters $\boldsymbol{\theta}^{\star}_{(T)}$

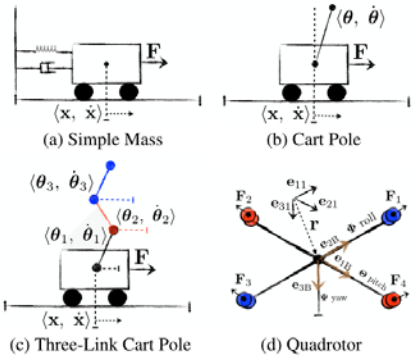
the robustness of the learned mapping by varying the number of source and target samples used for transfer and measuring the resultant target task performance. In all cases we compared the performance of MAXDT-PG to standard policy gradient learners. Our results show that MAXDT-PG was able to: *a*) learn a valid inter-state mapping with relatively little data from the target task, and *b*) effectively transfer between tasks from either the same or different domains.

Dynamical System Domains

We tested MAXDT-PG and standard policy gradient learning on four dynamical systems (Figure 2). On all systems, the reward function was based on two factors: *a*) penalizing states far from the goal state, and *b*) penalizing high forces (actions) to encourage smooth, low-energy movements.

Simple Mass Spring Damper (SM): The goal with the SM is to control the mass at a specified position with zero velocity. The system dynamics are described by two state-variables that represent the mass position and velocity, and a single force $\boldsymbol{F}$ that acts on the cart in the $\mathbf{x}$ direction.

Cart Pole (CP): The goal is to swing up and then balance the pole vertically. The system dynamics are described via a four-dimensional state vector $\langle \mathbf{x},\ \dot{\mathbf{x}},\ \boldsymbol{\theta},\ \dot{\boldsymbol{\theta}} \rangle$, representing the position, velocity of the cart, and the angle and angular velocity of the pole, respectively. The actions consist of a force that acts on the cart in the $\mathbf{x}$ direction.



(a) Simple Mass (b) Cart Pole

(c) Three-Link Cart Pole (d) Quadrotor

Figure 2: Dynamical systems used in the experiments.

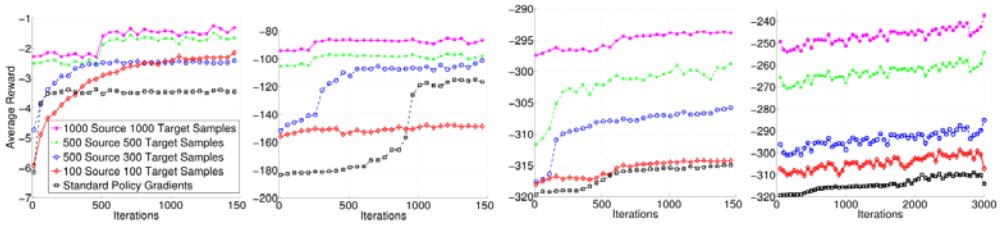
Three-Link Cart Pole (3CP): The 3CP dynamics are described via an eight-dimensional state vector $\langle \mathbf{x},\ \dot{\mathbf{x}},\ \boldsymbol{\theta}_1,\ \dot{\boldsymbol{\theta}}_1,\ \boldsymbol{\theta}_2,\ \dot{\boldsymbol{\theta}}_2,\ \boldsymbol{\theta}_3,\ \dot{\boldsymbol{\theta}}_3 \rangle$, where $\mathbf{x}$ and $\dot{\mathbf{x}}$ describe the position and velocity of the cart and $\boldsymbol{\theta}_j$ and $\dot{\boldsymbol{\theta}}_j$ represent the angle and angular velocity of the j^{th} link. The system is controlled by applying a force $\boldsymbol{F}$ to the cart in the $\mathbf{x}$ direction, with the goal of balancing the three poles upright.

Quadrotor (QR): The system dynamics were adopted from a simulator validated on real quadrotors (Bouabdallah 2007; Voos & Bou Ammar 2010), and are described via three angles and three angular velocities in the body frame (i.e., $\boldsymbol{e}_{1\mathrm{B}}$, $\boldsymbol{e}_{2\mathrm{B}}$, and $\boldsymbol{e}_{3\mathrm{B}}$). The actions consist of four rotor torques $\{\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3, \boldsymbol{F}_4\}$. Each task corresponds to a different quadrotor configuration (e.g., different armature lengths, etc.), and the goal is to stabilize the different quadrotors.

Same-Domain Transfer

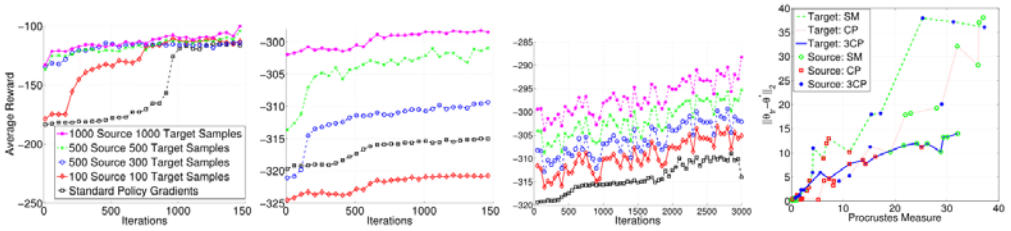
We first evaluate MAXDT-PG on same-domain transfer. Within each domain, we can obtain different tasks by varying the system parameters (e.g., for the SM system we varied mass M, spring constant K, and damping constant b) as well as the reward functions. We assessed the performance of using the transferred policy from MAXDT-PG versus standard policy gradients by measuring the average reward on the target task vs. the amount of learning iterations in the target. We also examined the robustness of MAXDT-PG's performance based on the number of source and target samples used to learn $\chi_{\mathcal{S}}$. Rewards were averaged over 500 traces collected from 150 initial states. Due to space constraints, we report same-domain transfer results here; details of the tasks and experimental procedure can be found in the appendix[2].

Figure 3 shows MAXDT-PG's performance using varying numbers of source and target samples to learn $\chi_{\mathcal{S}}$. These results reveal that transfer-initialized policies outperform standard policy gradient initialization. Further, as the number of samples used to learn $\chi_{\mathcal{S}}$ increases, so does both the initial and final performance in all domains. All initializations result in equal per-iteration computational cost. Therefore, MAXDT-PG both improves sample complexity and reduces wall-clock learning time.



(a) Simple Mass (b) Cart Pole (c) Three-Link Cart Pole (d) Quadrotor

Figure 3: Same-domain transfer results. All plots share the same legend and vertical axis label.



(a) Simple Mass to Cart Pole (b) Cart Pole to Three-Link CP (c) Cart Pole to Quadrotor (d) Alignment Quality vs Transfer

Figure 4: Cross-domain transfer results. Plots (a)–(c) depict target task performance, and share the same legend and axis labels. Plot (d) shows the correlation between manifold alignment quality (Procrustes metric) and quality of the transferred knowledge.

Cross-Domain Transfer

Next, we consider the more difficult problem of cross-domain transfer. The experimental setup is identical to the same-domain case with the crucial difference that the state and/or action spaces were different for the source and the target task (since the tasks were from different domains). We tested three cross-domain transfer scenarios: simple mass to cart pole, cart pole to three-link cart pole, and cart pole to quadrotor. In each case, the source and target task have different numbers of state variables and system dynamics. Details of these experiments are available in the appendix[2].

Figure 4 shows the results of cross-domain transfer, demonstrating that MAXDT-PG can achieve successful transfer between different task domains. These results reinforce the conclusions of the same-domain transfer experiments, showing that *a*) transfer-initialized policies outperform standard policy gradients, even between different task domains and *b*) initial and final performance improves as more samples are used to learn $\chi_{\mathcal{S}}$.

We also examined the correlation between the quality of the manifold alignment, as assessed by the Procrustes metric (Goldberg & Ritov 2009), and the quality of the transferred knowledge, as measured by the distance between the transferred ($\boldsymbol{\theta}_{\text{tr}}$) and the optimal ($\boldsymbol{\theta}^{\star}$) parameters (Figure 4(d)). On both measures, smaller values indicate better quality. Each data point represents a transfer scenario between two different tasks, from either SM, CP, or 3CP; we did not consider quadrotor tasks due to the required simulator time. Although we show that the Procrustes measure is positively correlated with transfer quality, we hesitate to recommend it as a predictive measure of transfer performance. In our approach, the cross-domain mapping is not guaranteed to be orthogonal, and therefore the Procrustes measure is not theoretically guaranteed to accurately measure the quality of the global embedding (i.e., Goldberg and Ritov's (2009) Corollary 1 is not guaranteed to hold), but the Procrustes measure still appears correlated with transfer quality in practice.

We can conclude that MAXDT-PG is capable of: *a*) automatically learning an inter-state mapping, and *b*) effectively transferring between different domain systems. Even when the source and target tasks are highly dissimilar (e.g., cart pole to quadrotor), MAXDT-PG is capable of successfully providing target policy initializations that outperform state-of-the-art policy gradient techniques.

Conclusion

We introduced MAXDT-PG, a technique for autonomous transfer between policy gradient RL algorithms. MAXDT-PG employs unsupervised manifold alignment to learn an inter-state mapping, which is then used to transfer samples and initialize the target task policy. MAXDT-PG's performance was evaluated on four dynamical systems, demonstrating that MAXDT-PG is capable of improving both an agent's initial and final performance relative to using policy gradient algorithms without transfer, even across different domains.

Acknowledgements

This research was supported by ONR grant #N00014-11-1-0139, AFOSR grant #FA8750-14-1-0069, and NSF grant IIS-1149917. We thank the anonymous reviewers for their helpful feedback.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Banerjee, B., and Stone, P. 2007. General game learning using knowledge transfer. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 672–677.

Bemporad, A.; Morari, M.; Dua, V.; and Pistikopoulos, E. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica* 38(1):3–20.

Bócsi, B.; Csato, L.; and Peters, J. 2013. Alignment-based transfer learning for robot models. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

Bou Ammar, H.; Taylor, M.; Tuyls, K.; Driessens, K.; and Weiss, G. 2012. Reinforcement learning transfer via sparse coding. In *Proceedings of the 11th Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Bouabdallah, S. 2007. *Design and Control of Quadrotors with Application to Autonomous Flying*. Ph.D. Dissertation, École polytechnique fédérale de Lausanne.

Buşoniu, L.; Babuška, R.; De Schutter, B.; and Ernst, D. 2010. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton, Florida: CRC Press.

Goldberg, Y.; and Ritov, Y. 2009. Local Procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Machine Learning* 77(1): 1–25.

Kober, J.; Bagnell, A.; and Peters, J. 2013. Reinforcement learning in robotics: a survey. *International Journal of Robotics Research* 32(11): 1238–1274.

Konidaris, G., and Barto, A. 2007. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 895–900.

Liu, Y., and Stone, P. 2006. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 415–20.

Peters, J., and Schaal, S. 2006. Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2219–2225.

Peters, J., and Schaal, S. 2008. Natural actor-critic. *Neurocomputing* 71(7-9):1180–1190.

Peters, J.; Vijayakumar, S.; and Schaal, S. 2005. Natural actor-critic. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, 280–291. Springer.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems* 1057–1063.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: a survey. *Journal of Machine Learning Research* 10:1633–1685.

Taylor, M. E.; Kuhlmann, G.; and Stone, P. 2008. Autonomous transfer for reinforcement learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 283–290.

Taylor, M. E.; Stone, P.; and Liu, Y. 2007. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research* 8(1):2125–2167.

Taylor, M. E.; Whiteson, S.; and Stone, P. 2007. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*.

Torrey, L.; Shavlik, J.; Walker, T.; and Maclin, R. 2008. Relational macros for transfer in reinforcement learning. In Blockeel, H.; Ramon, J.; Shavlik, J.; and Tadepalli, P., eds., *Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 254–268.

Voos, H., and Bou Ammar, H. 2010. Nonlinear tracking and landing controller for quadrotor aerial robots. In *Proceedings of the IEEE International Conference on Control Applications (CCA)*, 2136–2141.

Wang, C., and Mahadevan, S. 2009. Manifold alignment without correspondence. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1273–1278. Morgan Kaufmann.

Appendix C: Bou Aamar et al., 2015b

Safe Policy Search for Lifelong Reinforcement Learning with Sublinear Regret

Haitham Bou Ammar HAITHAMB@SEAS.UPENN.EDU
Rasul Tutunov TUTUNOV@SEAS.UPENN.EDU
Eric Eaton EEATON@CIS.UPENN.EDU
University of Pennsylvania, Computer and Information Science Department, Philadelphia, PA 19104 USA

Abstract

Lifelong reinforcement learning provides a promising framework for developing versatile agents that can accumulate knowledge over a lifetime of experience and rapidly learn new tasks by building upon prior knowledge. However, current lifelong learning methods exhibit non-vanishing regret as the amount of experience increases, and include limitations that can lead to suboptimal or unsafe control policies. To address these issues, we develop a lifelong policy gradient learner that operates in an adversarial setting to learn multiple tasks online while enforcing safety constraints on the learned policies. We demonstrate, for the first time, *sublinear regret* for lifelong policy search, and validate our algorithm on several benchmark dynamical systems and an application to quadrotor control.

1. Introduction

Reinforcement learning (RL) (Busoniu et al., 2010; Sutton & Barto, 1998) often requires substantial experience before achieving acceptable performance on individual control problems. One major contributor to this issue is the *tabula-rasa* assumption of typical RL methods, which learn from scratch on each new task. In these settings, learning performance is directly correlated with the quality of the acquired samples. Unfortunately, the amount of experience necessary for high-quality performance increases exponentially with the tasks' degrees of freedom, inhibiting the application of RL to high-dimensional control problems.

When data is in limited supply, transfer learning can significantly improve model performance on new tasks by reusing previous learned knowledge during training (Taylor & Stone, 2009; Gheshlaghi Azar et al., 2013; Lazaric, 2011; Ferrante et al., 2008; Bou Ammar et al., 2012). Multi-task learning (MTL) explores another notion of knowledge transfer, in which task models are trained simultaneously and share knowledge during the joint learning process (Wilson et al., 2007; Zhang et al., 2008).

In the *lifelong learning* setting (Thrun & O'Sullivan, 1996a;b), which can be framed as an online MTL problem, agents acquire knowledge incrementally by learning multiple tasks consecutively over their lifetime. Recently, based on the work of Ruvolo & Eaton (2013) on supervised lifelong learning, Bou Ammar et al. (2014) developed a lifelong learner for policy gradient RL. To ensure efficient learning over consecutive tasks, these works employ a second-order Taylor expansion around the parameters that are (locally) optimal for each task without transfer. This assumption simplifies the MTL objective into a weighted quadratic form for online learning, but since it is based on single-task learning, this technique can lead to parameters far from globally optimal. Consequently, the success of these methods for RL highly depends on the policy initializations, which must lead to near-optimal trajectories for meaningful updates. Also, since their objective functions average loss over all tasks, these methods exhibit non-vanishing regrets of the form $\mathcal{O}(R)$, where R is the total number of rounds in a non-adversarial setting.

In addition, these methods may produce control policies with unsafe behavior (i.e., capable of causing damage to the agent or environment, catastrophic failure, etc.). This is a critical issue in robotic control, where unsafe control policies can lead to physical damage or user injury. This problem is caused by using constraint-free optimization over the shared knowledge during the transfer process, which may lead to uninformative or unbounded policies.

In this paper, we address these issues by proposing the first *safe lifelong learner* for policy gradient RL operating in an adversarial framework. Our approach rapidly learns high-performance *safe control policies* based on the agent's previously learned knowledge and safety constraints on each task, accumulating knowledge over multiple consecutive tasks to optimize overall performance. We theoretically analyze the regret exhibited by our algorithm, showing *sublinear* dependency of the form $\mathcal{O}(\sqrt{R})$ for R rounds, thus outperforming current methods. We then evaluate our approach empirically on a set of dynamical systems.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

2. Background

2.1. Reinforcement Learning

An RL agent sequentially chooses actions to minimize its expected cost. Such problems are formalized as Markov decision processes (MDPs) $\langle \mathcal{X}, \mathcal{U}, \mathcal{P}, \boldsymbol{c}, \gamma \rangle$, where $\mathcal{X} \subset \mathbb{R}^d$ is the (potentially infinite) state space, $\mathcal{U} \in \mathbb{R}^{d_a}$ is the set of all possible actions, $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ is a state transition probability describing the system's dynamics, $\boldsymbol{c} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ is the cost function measuring the agent's performance, and $\gamma \in [0, 1]$ is a discount factor. At each time step m, the agent is in state $\boldsymbol{x}_m \in \mathcal{X}$ and must choose an action $\boldsymbol{u}_m \in \mathcal{U}$, transitioning it to a new state $\boldsymbol{x}_{m+1} \sim \mathcal{P}(\boldsymbol{x}_{m+1}|\boldsymbol{x}_m, \boldsymbol{u}_m)$ and yielding a cost $\boldsymbol{c}_{m+1} = \boldsymbol{c}(\boldsymbol{x}_{m+1}, \boldsymbol{u}_m, \boldsymbol{x}_m)$. The sequence of state-action pairs forms a trajectory $\boldsymbol{\tau} = [\boldsymbol{x}_{0:M-1}, \boldsymbol{u}_{0:M-1}]$ over a (possibly infinite) horizon M. A policy $\pi : \mathcal{X} \times \mathcal{U} \to [0, 1]$ specifies a probability distribution over state-action pairs, where $\pi(\boldsymbol{u}|\boldsymbol{x})$ represents the probability of selecting an action $\boldsymbol{u}$ in state $\boldsymbol{x}$. The goal of RL is to find an optimal policy $\pi^\star$ that minimizes the total expected cost.

Policy search methods have shown success in solving high-dimensional problems, such as robotic control (Kober & Peters, 2011; Peters & Schaal, 2008a; Sutton et al., 2000). These methods represent the policy $\pi_{\boldsymbol{\alpha}}(\boldsymbol{u}|\boldsymbol{x})$ using a vector $\boldsymbol{\alpha} \in \mathbb{R}^d$ of control parameters. The optimal policy $\pi^\star$ is found by determining the parameters $\boldsymbol{\alpha}^\star$ that minimize the expected average cost:

$$l(\boldsymbol{\alpha}) = \sum_{k=1}^{n} p_{\boldsymbol{\alpha}}\left(\boldsymbol{\tau}^{(k)}\right) \boldsymbol{C}\left(\boldsymbol{\tau}^{(k)}\right) \quad , \tag{1}$$

where n is the total number of trajectories, and $p_{\boldsymbol{\alpha}}\left(\boldsymbol{\tau}^{(k)}\right)$ and $\boldsymbol{C}\left(\boldsymbol{\tau}^{(k)}\right)$ are the probability and cost of trajectory $\boldsymbol{\tau}^{(k)}$:

$$p_{\boldsymbol{\alpha}}\left(\boldsymbol{\tau}^{(k)}\right) = \mathcal{P}_0\left(\boldsymbol{x}_0^{(k)}\right) \prod_{m=0}^{M-1} \mathcal{P}\left(\boldsymbol{x}_{m+1}^{(k)}|\boldsymbol{x}_m^{(k)}, \boldsymbol{u}_m^{(k)}\right) \times \pi_{\boldsymbol{\alpha}}\left(\boldsymbol{u}_m^{(k)}|\boldsymbol{x}_m^{(k)}\right) \tag{2}$$

$$\boldsymbol{C}\left(\boldsymbol{\tau}^{(k)}\right) = \frac{1}{M} \sum_{m=0}^{M-1} \boldsymbol{c}\left(\boldsymbol{x}_{m+1}^{(k)}, \boldsymbol{u}_m^{(k)}, \boldsymbol{x}_m^{(k)}\right) \quad , \tag{3}$$

with an initial state distribution $\mathcal{P}_0 : \mathcal{X} \to [0, 1]$. We handle a constrained version of policy search, in which optimality not only corresponds to minimizing the total expected cost, but also to ensuring that the policy satisfies safety constraints. These constraints vary between applications, for example corresponding to maximum joint torque or prohibited physical positions.

2.2. Online Learning & Regret Analysis

In this paper, we employ a special form of *regret minimization games*, which we briefly review here. A regret minimization game is a triple $\langle \mathcal{K}, \mathcal{F}, R \rangle$, where $\mathcal{K}$ is a non-empty decision set, $\mathcal{F}$ is the set of moves of the adversary which contains bounded convex functions from $\mathbb{R}^n$ to $\mathbb{R}$, and R is the total number of rounds. The game proceeds in rounds, where at each round $j = 1, \ldots, R$, the agent chooses a prediction $\boldsymbol{\theta}_j \in \mathcal{K}$ and the environment (i.e., the adversary) chooses a loss function $l_j \in \mathcal{F}$. At the end of the round, the loss function l_j is revealed to the agent and the decision $\boldsymbol{\theta}_j$ is revealed to the environment. In this paper, we handle the full-information case, where the agent may observe the entire loss function l_j as its feedback and can exploit this in making decisions. The goal is to minimize the cumulative regret $\sum_{j=1}^{R} l_j(\boldsymbol{\theta}_j) - \inf_{\boldsymbol{u} \in \mathcal{K}} \left[\sum_{j=1}^{R} l_j(\boldsymbol{u})\right]$. When analyzing the regret of our methods, we use a variant of this definition to handle the lifelong RL case:

$$\mathfrak{R}_R = \sum_{j=1}^{R} l_{t_j}(\boldsymbol{\theta}_j) - \inf_{u \in \mathcal{K}} \left[\sum_{j=1}^{R} l_{t_j}(\boldsymbol{u})\right] \quad ,$$

where $l_{t_j}(\cdot)$ denotes the loss of task t at round j.

For our framework, we adopt a variant of regret minimization called "Follow the Regularized Leader," which minimizes regret in two steps. First, the unconstrained solution $\tilde{\boldsymbol{\theta}}$ is determined (see Sect. 4.1) by solving an unconstrained optimization over the accumulated losses observed so far. Given $\tilde{\boldsymbol{\theta}}$, the constrained solution is then determined by learning a projection into the constraint set via Bregman projections (see Abbasi-Yadkori et al. (2013)).

3. Safe Lifelong Policy Search

We adopt a lifelong learning framework in which the agent learns multiple RL tasks consecutively, providing it the opportunity to transfer knowledge between tasks to improve learning. Let $\mathcal{T}$ denote the set of tasks, each element of which is an MDP. At any time, the learner may face any previously seen task, and so must strive to maximize its performance across all tasks. The goal is to learn optimal policies $\pi^\star_{\boldsymbol{\alpha}_1^\star}, \ldots, \pi^\star_{\boldsymbol{\alpha}_{|\mathcal{T}|}^\star}$ for all tasks, where policy $\pi^\star_{\boldsymbol{\alpha}_t^\star}$ for task t is parameterized by $\boldsymbol{\alpha}_t^\star \in \mathbb{R}^d$. In addition, each task is equipped with safety constraints to ensure acceptable policy behavior: $\boldsymbol{A}_t \boldsymbol{\alpha}_t \leq \boldsymbol{b}_t$, with $\boldsymbol{A}_t \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b}_t \in \mathbb{R}^d$ representing the allowed policy combinations. The precise form of these constraints depends on the application domain, but this formulation supports constraints on (e.g.) joint torque, acceleration, position, etc.

At each round j, the learner observes a set of n_{t_j} trajectories $\left\{\boldsymbol{\tau}_{t_j}^{(1)}, \ldots, \boldsymbol{\tau}_{t_j}^{(n_{t_j})}\right\}$ from a task $t_j \in \mathcal{T}$, where each trajectory has length M_{t_j}. To support knowledge transfer between tasks, we assume that each task's policy parameters $\boldsymbol{\alpha}_{t_j} \in \mathbb{R}^d$ at round j can be written as a linear combination of a shared latent basis $\boldsymbol{L} \in \mathbb{R}^{d \times k}$ with coefficient vectors $\boldsymbol{s}_{t_j} \in \mathbb{R}^k$; therefore, $\boldsymbol{\alpha}_{t_j} = \boldsymbol{L}\boldsymbol{s}_{t_j}$. Each column of $\boldsymbol{L}$ represents a chunk of transferrable knowledge; this task construction has been used successfully in previous

multi-task learning work (Kumar & Daumé III, 2012; Ruvolo & Eaton, 2013; Bou Ammar et al., 2014). Extending this previous work, we ensure that the shared knowledge repository is "informative" by incorporating bounding constraints on the Frobenius norm $\|\cdot\|_{\mathsf{F}}$ of $\boldsymbol{L}$. Consequently, the optimization problem after observing r rounds is:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \sum_{j=1}^{r} \left[\eta_{t_j} l_{t_j}\left(\boldsymbol{L}\boldsymbol{s}_{t_j}\right)\right] + \mu_1 \|\boldsymbol{S}\|_{\mathsf{F}}^2 + \mu_2 \|\boldsymbol{L}\|_{\mathsf{F}}^2 \qquad (4)$$
$$\text{s.t.} \quad \boldsymbol{A}_{t_j}\boldsymbol{\alpha}_{t_j} \leq \boldsymbol{b}_{t_j} \;\; \forall t_j \in \mathcal{I}_r$$
$$\boldsymbol{\lambda}_{\min}\left(\boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}\right) \geq p \;\text{ and }\; \boldsymbol{\lambda}_{\max}\left(\boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}\right) \leq q \;,$$

where p and q are the constraints on $\|\boldsymbol{L}\|_{\mathsf{F}}$, $\eta_{t_j} \in \mathbb{R}$ are design weighting parameters[1], $\mathcal{I}_r = \{t_1, \ldots, t_r\}$ denotes the set of all tasks observed so far through round r, and $\boldsymbol{S}$ is the collection of all coefficients

$$\boldsymbol{S}(:,h) = \begin{cases} \boldsymbol{s}_{t_h} & \text{if } t_h \in \mathcal{I}_r \\ 0 & \text{otherwise} \end{cases} \quad \forall h \in \{1, \ldots, |\mathcal{T}|\} \;.$$

The loss function $l_{t_j}(\boldsymbol{\alpha}_{t_j})$ in Eq. (4) corresponds to a policy gradient learner for task t_j, as defined in Eq. (1). Typical policy gradient methods (Kober & Peters, 2011; Sutton et al., 2000) maximize a lower bound of the expected cost $l_{t_j}(\boldsymbol{\alpha}_{t_j})$, which can be derived by taking the logarithm and applying Jensen's inequality:

$$\log\left[l_{t_j}(\boldsymbol{\alpha}_{t_j})\right] = \log\left[\sum_{k=1}^{n_{t_j}} p_{\boldsymbol{\alpha}_{t_j}}^{(t_j)}\left(\boldsymbol{\tau}_{t_j}^{(k)}\right) \boldsymbol{C}^{(t_j)}\left(\boldsymbol{\tau}_{t_j}^{(k)}\right)\right] \qquad (5)$$
$$\geq \log\left[n_{t_j}\right] + \mathbb{E}\left[\sum_{m=0}^{M_{t_j}-1} \log\left[\pi_{\boldsymbol{\alpha}_{t_j}}\left(\boldsymbol{u}_m^{(k,t_j)} \mid \boldsymbol{x}_m^{(k,t_j)}\right)\right]\right]_{k=1}^{n_{t_j}} + \text{const} \;.$$

Therefore, our goal is to minimize the following objective:

$$\boldsymbol{e}_r = \sum_{j=1}^{r}\left(-\frac{\eta_{t_j}}{n_{t_j}} \sum_{k=1}^{n_{t_j}} \sum_{m=0}^{M_{t_j}-1} \log\left[\pi_{\boldsymbol{\alpha}_{t_j}}\left(\boldsymbol{u}_m^{(k,t_j)} \mid \boldsymbol{x}_m^{(k,t_j)}\right)\right]\right) \qquad (6)$$
$$+ \mu_1 \|\boldsymbol{S}\|_{\mathsf{F}}^2 + \mu_2 \|\boldsymbol{L}\|_{\mathsf{F}}^2$$
$$\text{s.t.} \quad \boldsymbol{A}_{t_j}\boldsymbol{\alpha}_{t_j} \leq \boldsymbol{b}_{t_j} \;\; \forall t_j \in \mathcal{I}_r$$
$$\boldsymbol{\lambda}_{\min}\left(\boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}\right) \geq p \;\text{ and }\; \boldsymbol{\lambda}_{\max}\left(\boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}\right) \leq q \;.$$

3.1. Online Formulation

The optimization problem above can be mapped to the standard online learning framework by unrolling $\boldsymbol{L}$ and $\boldsymbol{S}$ into a vector $\boldsymbol{\theta} = [\text{vec}(\boldsymbol{L})\ \text{vec}(\boldsymbol{S})]^{\mathsf{T}} \in \mathbb{R}^{dk+k|\mathcal{T}|}$. Choosing $\boldsymbol{\Omega}_0(\boldsymbol{\theta}) = \mu_2 \sum_{i=1}^{dk} \boldsymbol{\theta}_i^2 + \mu_1 \sum_{i=dk+1}^{dk+k|\mathcal{T}|} \boldsymbol{\theta}_i^2$, and $\boldsymbol{\Omega}_j(\boldsymbol{\theta}) = \boldsymbol{\Omega}_{j-1}(\boldsymbol{\theta}) + \eta_{t_j} l_{t_j}(\boldsymbol{\theta})$, we can write the safe lifelong policy search problem (Eq. (6)) as:

$$\boldsymbol{\theta}_{r+1} = \arg\min_{\boldsymbol{\theta} \in \mathcal{K}} \boldsymbol{\Omega}_r(\boldsymbol{\theta}) \;, \qquad (7)$$

where $\mathcal{K} \subseteq \mathbb{R}^{dk+k|\mathcal{T}|}$ is the set of allowable policies under the given safety constraints. Note that the loss for task t_j can be written as a bilinear product in $\boldsymbol{\theta}$:

$$l_{t_j}(\boldsymbol{\theta}) = -\frac{1}{n_{t_j}} \sum_{k=1}^{n_{t_j}} \sum_{m=0}^{M_{t_j}-1} \log\left[\pi_{\boldsymbol{\Theta}_{\boldsymbol{L}}\boldsymbol{\Theta}_{\boldsymbol{s}_{t_j}}}\left(\boldsymbol{u}_m^{(k,\,t_j)} \mid \boldsymbol{x}_m^{(k,\,t_j)}\right)\right]$$

$$\boldsymbol{\Theta}_{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{\theta}_1 & \ldots & \boldsymbol{\theta}_{d(k-1)+1} \\ \vdots & \vdots & \vdots \\ \boldsymbol{\theta}_d & \ldots & \boldsymbol{\theta}_{dk} \end{bmatrix}, \; \boldsymbol{\Theta}_{\boldsymbol{s}_{t_j}} = \begin{bmatrix} \boldsymbol{\theta}_{dk+1} \\ \vdots \\ \boldsymbol{\theta}_{(d+1)k+1} \end{bmatrix}.$$

We see that the problem in Eq. (7) is equivalent to Eq. (6) by noting that at r rounds, $\boldsymbol{\Omega}_r = \sum_{j=1}^{r} \eta_{t_j} l_{t_j}(\boldsymbol{\theta}) + \boldsymbol{\Omega}_0(\boldsymbol{\theta})$.

4. Online Learning Method

We solve Eq. (7) in two steps. First, we determine the unconstrained solution $\tilde{\boldsymbol{\theta}}_{r+1}$ when $\mathcal{K} = \mathbb{R}^{dk+k|\mathcal{T}|}$ (see Sect. 4.1). Given $\tilde{\boldsymbol{\theta}}_{r+1}$, we derive the constrained solution $\hat{\boldsymbol{\theta}}_{r+1}$ by learning a projection $\text{Proj}_{\boldsymbol{\Omega}_r,\mathcal{K}}\left(\tilde{\boldsymbol{\theta}}_{r+1}\right)$ to the constraint set $\mathcal{K} \subseteq \mathbb{R}^{dk+k|\mathcal{T}|}$, which amounts to minimizing the Bregman divergence over $\boldsymbol{\Omega}_r(\boldsymbol{\theta})$ (see Sect. 4.2)[2]. The complete approach is given in Algorithm 1 and is available as a software implementation on the authors' websites.

4.1. Unconstrained Policy Solution

Although Eq. (6) is not jointly convex in both $\boldsymbol{L}$ and $\boldsymbol{S}$, it is separably convex (for log-concave policy distributions). Consequently, we follow an alternating optimization approach, first computing $\boldsymbol{L}$ while holding $\boldsymbol{S}$ fixed, and then updating $\boldsymbol{S}$ given the acquired $\boldsymbol{L}$. We detail this process for two popular PG learners, eREINFORCE (Williams, 1992) and eNAC (Peters & Schaal, 2008b). The derivations of the update rules below can be found in Appendix A.

These updates are governed by learning rates β and λ that decay over time; β and λ can be chosen using line-search methods as discussed by Boyd & Vandenberghe (2004). In our experiments, we adopt a simple yet effective strategy, where $\beta = cj^{-1}$ and $\lambda = cj^{-1}$, with $0 < c < 1$.

Step 1: Updating $\boldsymbol{L}$ Holding $\boldsymbol{S}$ fixed, the latent repository can be updated according to:

$$\boldsymbol{L}_{\beta+1} = \boldsymbol{L}_{\beta} - \eta_{\boldsymbol{L}}^{\beta} \nabla_{\boldsymbol{L}} \boldsymbol{e}_r(\boldsymbol{L}, \boldsymbol{S}) \qquad \text{(eREINFORCE)}$$
$$\boldsymbol{L}_{\beta+1} = \boldsymbol{L}_{\beta} - \eta_{\boldsymbol{L}}^{\beta} \boldsymbol{G}^{-1}(\boldsymbol{L}_{\beta}, \boldsymbol{S}_{\beta}) \nabla_{\boldsymbol{L}} \boldsymbol{e}_r(\boldsymbol{L}, \boldsymbol{S}) \qquad \text{(eNAC)}$$

with learning rate $\eta_{\boldsymbol{L}}^{\beta} \in \mathbb{R}$, and $\boldsymbol{G}^{-1}(\boldsymbol{L}, \boldsymbol{S})$ as the inverse of the Fisher information matrix (Peters & Schaal, 2008b).

In the special case of Gaussian policies, the update for $\boldsymbol{L}$

[1] We describe later how to set the η's later in Sect. 5 to obtain regret bounds, and leave them as variables now for generality.

[2] In Sect. 4.2, we linearize the loss around the constrained solution of the previous round to increase stability and ensure convergence. Given the linear losses, it suffices to solve the Bregman divergence over the regularizer, reducing the computational cost.

can be derived in a closed form as $\boldsymbol{L}_{\beta+1} = \boldsymbol{Z}_{\boldsymbol{L}}^{-1}\boldsymbol{v}_{\boldsymbol{L}}$, where

$$\boldsymbol{Z}_{\boldsymbol{L}} = 2\mu_2 \boldsymbol{I}_{dk\times dk} + \sum_{j=1}^{r} \frac{\eta_{t_j}}{n_{t_j}\sigma_{t_j}^2} \sum_{k=1}^{n_{t_j}} \sum_{m=0}^{M_{t_j}-1} \mathrm{vec}\left(\boldsymbol{\Phi}\boldsymbol{s}_{t_j}^\mathsf{T}\right)\left(\boldsymbol{\Phi}^\mathsf{T} \otimes \boldsymbol{s}_{t_j}^\mathsf{T}\right)$$

$$\boldsymbol{v}_{\boldsymbol{L}} = \sum_{j} \frac{\eta_{t_j}}{n_{t_j}\sigma_{t_j}^2} \sum_{k=1}^{n_{t_j}} \sum_{m=0}^{M_{t_j}-1} \mathrm{vec}\left(\boldsymbol{u}_m^{(k,\,t_j)} \boldsymbol{\Phi}\boldsymbol{s}_{t_j}^\mathsf{T}\right) \ ,$$

$\sigma_{t_j}^2$ is the covariance of the Gaussian policy for a task t_j, and $\boldsymbol{\Phi} = \boldsymbol{\Phi}\left(\boldsymbol{x}_m^{(k,\,t_j)}\right)$ denotes the state features.

Step 2: Updating $\boldsymbol{S}$ Given the fixed basis $\boldsymbol{L}$, the coefficient matrix $\boldsymbol{S}$ is updated column-wise for all $t_j \in \mathcal{I}_r$:

$$\boldsymbol{s}_{\lambda+1}^{(t_j)} = \boldsymbol{s}_{\lambda+1}^{(t_j)} - \eta_{\boldsymbol{S}}^{\lambda} \nabla_{\boldsymbol{s}_{t_j}} e_r(\boldsymbol{L}, \boldsymbol{S}) \qquad \text{(eREINFORCE)}$$

$$\boldsymbol{s}_{\lambda+1}^{(t_j)} = \boldsymbol{s}_{\lambda+1}^{(t_j)} - \eta_{\boldsymbol{S}}^{\lambda} \boldsymbol{G}^{-1}(\boldsymbol{L}_\beta, \boldsymbol{S}_\beta) \nabla_{\boldsymbol{s}_{t_j}} e_r(\boldsymbol{L}, \boldsymbol{S}) \qquad \text{(eNAC)}$$

with learning rate $\eta_{\boldsymbol{S}}^{\lambda} \in \mathbb{R}$. For Gaussian policies, the closed-form of the update is $\boldsymbol{s}_{t_j} = \boldsymbol{Z}_{\boldsymbol{s}_{t_j}}^{-1}\boldsymbol{v}_{\boldsymbol{s}_{t_j}}$, where

$$\boldsymbol{Z}_{\boldsymbol{s}_{t_j}} = 2\mu_1 \boldsymbol{I}_{k\times k} + \sum_{t_k=t_j} \frac{\eta_{t_j}}{n_{t_j}\sigma_{t_j}^2} \sum_{k=1}^{n_{t_j}} \sum_{m=0}^{M_{t_j}-1} \boldsymbol{L}^\mathsf{T}\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{L}$$

$$\boldsymbol{v}_{t_j} = \sum_{t_k=t_j} \frac{\eta_{t_j}}{n_{t_j}\sigma_{t_j}^2} \sum_{k=1}^{n_{t_j}} \sum_{m=0}^{M_{t_j}-1} \boldsymbol{u}_m^{(k,\,t_j)} \boldsymbol{L}^\mathsf{T}\boldsymbol{\Phi} \ .$$

4.2. Constrained Policy Solution

Once we have obtained the unconstrained solution $\tilde{\boldsymbol{\theta}}_{r+1}$ (which satisfies Eq. (7), but can lead to policy parameters in unsafe regions), we then derive the constrained solution to ensure safe policies. We learn a projection $\mathrm{Proj}_{\boldsymbol{\Omega}_r,\mathcal{K}}\left(\tilde{\boldsymbol{\theta}}_{r+1}\right)$ from $\tilde{\boldsymbol{\theta}}_{r+1}$ to the constraint set:

$$\hat{\boldsymbol{\theta}}_{r+1} = \arg\min_{\boldsymbol{\theta}\in\mathcal{K}} \mathcal{B}_{\boldsymbol{\Omega}_r,\mathcal{K}}\left(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_{r+1}\right) \ , \qquad (8)$$

where $\mathcal{B}_{\boldsymbol{\Omega}_r,\mathcal{K}}\left(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_{r+1}\right)$ is the Bregman divergence over $\boldsymbol{\Omega}_r$:

$$\mathcal{B}_{\boldsymbol{\Omega}_r,\mathcal{K}}\left(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_{r+1}\right) = \boldsymbol{\Omega}_r(\boldsymbol{\theta}) - \boldsymbol{\Omega}_r(\tilde{\boldsymbol{\theta}}_{r+1}) - \mathrm{trace}\left(\nabla_{\boldsymbol{\theta}}\boldsymbol{\Omega}_r\left(\boldsymbol{\theta}\right)\Big|_{\tilde{\boldsymbol{\theta}}_{r+1}}\left(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}_{r+1}\right)\right) .$$

Solving Eq. (8) is computationally expensive since $\boldsymbol{\Omega}_r(\boldsymbol{\theta})$ includes the sum back to the original round. To remedy this problem, ensure the stability of our approach, and guarantee that the constrained solutions for all observed tasks lie within a bounded region, we linearize the current-round loss function $l_{t_r}(\boldsymbol{\theta})$ around the *constrained* solution of the previous round $\hat{\boldsymbol{\theta}}_r$:

$$l_{t_r}(\hat{\boldsymbol{u}}) = \hat{\boldsymbol{f}}_{t_r}\Big|_{\hat{\boldsymbol{\theta}}_r}^{\mathsf{T}} \hat{\boldsymbol{u}} \ , \qquad (9)$$

where

$$\hat{\boldsymbol{f}}_{t_r}\Big|_{\hat{\boldsymbol{\theta}}_r} = \begin{bmatrix} \nabla_{\boldsymbol{\theta}} l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r} \\ l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r} - \nabla_{\boldsymbol{\theta}} l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\hat{\boldsymbol{\theta}}_r \end{bmatrix} , \quad \hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix} .$$

Given the above linear form, we can rewrite the optimization problem in Eq. (8) as:

$$\hat{\boldsymbol{\theta}}_{r+1} = \arg\min_{\boldsymbol{\theta}\in\mathcal{K}} \mathcal{B}_{\boldsymbol{\Omega}_0,\mathcal{K}}\left(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_{r+1}\right) \ . \qquad (10)$$

Consequently, determining *safe policies* for lifelong policy search reinforcement learning amounts to solving:

$$\begin{aligned} \min_{\boldsymbol{L},\boldsymbol{S}} \ & \mu_1\|\boldsymbol{S}\|_{\mathsf{F}}^2 + \boldsymbol{\mu}_2\|\boldsymbol{L}\|_{\mathsf{F}}^2 \\ & + 2\mu_1 \mathrm{trace}\left(\boldsymbol{S}^\mathsf{T}\Big|_{\tilde{\theta}_{r+1}}\boldsymbol{S}\right) + 2\mu_2 \mathrm{trace}\left(\boldsymbol{L}\Big|_{\tilde{\theta}_{r+1}}\boldsymbol{L}\right) \\ \text{s.t. } & \boldsymbol{A}_{t_j}\boldsymbol{L}\boldsymbol{s}_{t_j} \le \boldsymbol{b}_{t_j} \quad \forall t_j \in \mathcal{I}_r \\ & \boldsymbol{L}\boldsymbol{L}^\mathsf{T} \le p\boldsymbol{I} \quad \text{and} \quad \boldsymbol{L}\boldsymbol{L}^\mathsf{T} \ge q\boldsymbol{I} \ . \end{aligned}$$

To solve the optimization problem above, we start by converting the inequality constraints to equality constraints by introducing slack variables $\boldsymbol{c}_{t_j} \ge 0$. We also guarantee that these slack variables are bounded by incorporating $\|\boldsymbol{c}_{t_j}\| \le \boldsymbol{c}_{\max},\ \forall t_j \in \{1,\dots,|\mathcal{T}|\}$:

$$\begin{aligned} \min_{\boldsymbol{L},\boldsymbol{S},\boldsymbol{C}} \ & \mu_1\|\boldsymbol{S}\|_{\mathsf{F}}^2 + \mu_2\|\boldsymbol{L}\|_{\mathsf{F}}^2 \\ & + 2\mu_2 \mathrm{trace}\left(\boldsymbol{L}^\mathsf{T}\Big|_{\tilde{\theta}_{r+1}}\boldsymbol{L}\right) + 2\mu_1 \mathrm{trace}\left(\boldsymbol{S}^\mathsf{T}\Big|_{\tilde{\theta}_{r+1}}\boldsymbol{S}\right) \\ \text{s.t. } & \boldsymbol{A}_{t_j}\boldsymbol{L}\boldsymbol{s}_{t_j} = \boldsymbol{b}_{t_j} - \boldsymbol{c}_{t_j} \quad \forall t_j \in \mathcal{I}_r \\ & \boldsymbol{c}_{t_j} > 0 \quad \text{and} \quad \|\boldsymbol{c}_{t_j}\|_2 \le \boldsymbol{c}_{\max} \quad \forall t_j \in \mathcal{I}_r \\ & \boldsymbol{L}\boldsymbol{L}^\mathsf{T} \le p\boldsymbol{I} \quad \text{and} \quad \boldsymbol{L}\boldsymbol{L}^\mathsf{T} \ge q\boldsymbol{I} \ . \end{aligned}$$

With this formulation, learning $\mathrm{Proj}_{\boldsymbol{\Omega}_r,\mathcal{K}}\left(\tilde{\boldsymbol{\theta}}_{r+1}\right)$ amounts to solving second-order cone and semi-definite programs.

4.2.1. Semi-Definite Program for Learning $\boldsymbol{L}$

This section determines the constrained projection of the shared basis $\boldsymbol{L}$ given fixed $\boldsymbol{S}$ and $\boldsymbol{C}$. We show that $\boldsymbol{L}$ can be acquired efficiently, since this step can be relaxed to solving a semi-definite program in $\boldsymbol{L}\boldsymbol{L}^\mathsf{T}$ (Boyd & Vandenberghe, 2004). To formulate the semi-definite program, note that

$$\begin{aligned} \mathrm{trace}\left(\boldsymbol{L}^\mathsf{T}\Big|_{\tilde{\theta}_{r+1}}\boldsymbol{L}\right) &= \sum_{i=1}^{k} \boldsymbol{l}_{r+1}^{(i)\,\mathsf{T}}\Big|_{\tilde{\theta}_{r+1}}\boldsymbol{l}_i \\ &\le \sum_{i=1}^{k} \left\|\boldsymbol{l}_{r+1}^{(i)}\Big|_{\tilde{\theta}_{r+1}}\right\|_2 \|\boldsymbol{l}_i\|_2 \\ &\le \sqrt{\sum_{i=1}^{k}\left\|\boldsymbol{l}_r^{(i)}\Big|_{\tilde{\theta}_{r+1}}\right\|_2^2}\sqrt{\sum_{i=1}^{k}\|\boldsymbol{l}_i\|_2^2} \\ &= \left\|\boldsymbol{L}\Big|_{\tilde{\theta}_{r+1}}\right\|_{\mathsf{F}} \sqrt{\mathrm{trace}\left(\boldsymbol{L}\boldsymbol{L}^\mathsf{T}\right)} \ . \end{aligned}$$

From the constraint set, we recognize:

$$\begin{aligned} & \boldsymbol{s}_{t_j}^\mathsf{T}\boldsymbol{L}^\mathsf{T} = \left(\boldsymbol{b}_{t_j} - \boldsymbol{c}_{t_j}\right)^\mathsf{T}\left(\boldsymbol{A}_{t_j}^\dagger\right)^\mathsf{T} \\ \Longrightarrow \ & \boldsymbol{s}_{t_j}^\mathsf{T}\boldsymbol{L}^\mathsf{T}\boldsymbol{L}\boldsymbol{s}_{t_j} = \boldsymbol{a}_{t_j}^\mathsf{T}\boldsymbol{a}_{t_j} \quad \text{with} \quad \boldsymbol{a}_{t_j} = \boldsymbol{A}_{t_j}^\dagger\left(\boldsymbol{b}_{t_j} - \boldsymbol{c}_{t_j}\right) \ . \end{aligned}$$

Algorithm 1 Safe Online Lifelong Policy Search

Inputs: Total number of rounds R, weighting factor $\eta = 1/\sqrt{R}$, regularization parameters μ_1 and μ_2, constraints p and q, number of latent basis vectors k.
$\boldsymbol{S} = \text{zeros}(k, |\mathcal{T}|)$, $\boldsymbol{L} = \text{diag}_k(\zeta)$ with $p \leq \zeta^2 \leq q$
for $j = 1$ to R **do**
  $t_j \leftarrow$ sampleTask(), and update $\mathcal{I}_j$
  Compute **unconstrained solution** $\tilde{\boldsymbol{\theta}}_{j+1}$ (Sect. 4.1)
  Fix $\boldsymbol{S}$ and $\boldsymbol{C}$, and update $\boldsymbol{L}$ (Sect. 4.2.1)
  Use updated $\boldsymbol{L}$ to derive $\boldsymbol{S}$ and $\boldsymbol{C}$ (Sect. 4.2.2)
end for
Output: Safety-constrained $\boldsymbol{L}$ and $\boldsymbol{S}$

Since spectrum $\left(\boldsymbol{L}\boldsymbol{L}^\top\right) =$ spectrum $\left(\boldsymbol{L}^\top\boldsymbol{L}\right)$, we can write:

$$\min_{\boldsymbol{X}\subset\mathcal{S}_{++}} \mu_2 \text{trace}(\boldsymbol{X}) + 2\mu_2 \left\| \boldsymbol{L}\big|_{\hat{\boldsymbol{\theta}}_{r+1}} \right\|_{\text{F}} \sqrt{\text{trace}\left(\boldsymbol{X}\right)}$$
$$\text{s.t. } \boldsymbol{s}_{t_j}^\top \boldsymbol{X} \boldsymbol{s}_{t_j} = \boldsymbol{a}_{t_j}^\top \boldsymbol{a}_{t_j} \quad \forall t_j \in \mathcal{I}_r$$
$$\boldsymbol{X} \leq p\boldsymbol{I} \text{ and } \boldsymbol{X} \geq q\boldsymbol{I}, \text{ with } \boldsymbol{X} = \boldsymbol{L}^\top\boldsymbol{L}.$$

4.2.2. Second-Order Cone Program for Learning Task Projections

Having determined $\boldsymbol{L}$, we can acquire $\boldsymbol{S}$ and update $\boldsymbol{C}$ by solving a second-order cone program (Boyd & Vandenberghe, 2004) of the following form:

$$\min_{\boldsymbol{s}_{t_1},\dots,\boldsymbol{s}_{t_j},\boldsymbol{c}_{t_1},\dots,\boldsymbol{c}_{t_j}} \mu_1 \sum_{j=1}^{r} \|\boldsymbol{s}_{t_j}\|_2^2 + 2\mu_1 \sum_{j=1}^{r} \boldsymbol{s}_{t_j}^\top\big|_{\hat{\boldsymbol{\theta}}_r} \boldsymbol{s}_{t_j}$$
$$\text{s.t. } \boldsymbol{A}_{t_j}\boldsymbol{L}\boldsymbol{s}_{t_j} = \boldsymbol{b}_{t_j} - \boldsymbol{c}_{t_j}$$
$$\boldsymbol{c}_{t_j} > 0 \quad \|\boldsymbol{c}_{t_j}\|_2^2 \leq \boldsymbol{c}_{\max}^2 \; \forall t_j \in \mathcal{I}_r \; .$$

5. Theoretical Guarantees

This section quantifies the performance of our approach by providing formal analysis of the regret after R rounds. We show that the safe lifelong reinforcement learner exhibits *sublinear* regret in the total number of rounds. Formally, we prove the following theorem:

Theorem 1 (Sublinear Regret). *After R rounds and choosing $\forall t_j \in \mathcal{I}_R \;\; \eta_{t_j} = \eta = \frac{1}{\sqrt{R}}$, $\boldsymbol{L}\big|_{\hat{\boldsymbol{\theta}}_1} = diag_{\boldsymbol{k}}(\zeta)$, with $diag_{\boldsymbol{k}}(\cdot)$ being a diagonal matrix among the $\boldsymbol{k}$ columns of $\boldsymbol{L}$, $p \leq \zeta^2 \leq q$, and $\boldsymbol{S}\big|_{\hat{\boldsymbol{\theta}}_1} = \boldsymbol{0}_{k\times|\mathcal{T}|}$, the safe lifelong reinforcement learner exhibits sublinear regret of the form:*

$$\sum_{j=1}^{R} l_{t_j}\left(\hat{\boldsymbol{\theta}}_j\right) - l_{t_j}(\boldsymbol{u}) = \mathcal{O}\left(\sqrt{R}\right) \text{ for any } \boldsymbol{u} \in \mathcal{K}.$$

Proof Roadmap: The remainder of this section completes our proof of Theorem 1; further details are given in Appendix B. We assume linear losses for all tasks in the constrained case in accordance with Sect. 4.2. Although linear losses for policy search RL are too restrictive given a single operating point, as discussed previously, we remedy this problem by generalizing to the case of piece-wise linear losses, where the linearization operating point is a resultant of the optimization problem. To bound the regret, we need to bound the dual Euclidean norm (which is the same as the Euclidean norm) of the gradient of the loss function, then prove Theorem 1 by bounding: (1) task t_j's gradient loss (Sect. 5.1), and (2) linearized losses with respect to $\boldsymbol{L}$ and $\boldsymbol{S}$ (Sect. 5.2).

5.1. Bounding t_j's Gradient Loss

We start by stating essential lemmas for Theorem 1; due to space constraints, proofs for all lemmas are available in the supplementary material. Here, we bound the gradient of a loss function $l_{t_j}(\boldsymbol{\theta})$ at round r under Gaussian policies[3].

Assumption 1. *We assume that the policy for a task t_j is Gaussian, the action set $\mathcal{U}$ is bounded by $\boldsymbol{u}_{\max}$, and the feature set is upper-bounded by $\boldsymbol{\Phi}_{\max}$.*

Lemma 1. *Assume task t_j's policy at round r is given by $\pi_{\boldsymbol{\alpha}_{t_j}}^{(t_j)}\left(\boldsymbol{u}_m^{(k,\,t_j)}|\boldsymbol{x}_m^{(k,\,t_j)}\right)\Big|_{\hat{\boldsymbol{\theta}}_r} = \mathcal{N}\left(\boldsymbol{\alpha}_{t_j}^\top\big|_{\hat{\boldsymbol{\theta}}_r}\boldsymbol{\Phi}\left(\boldsymbol{x}_m^{(k,\,t_j)}\right), \boldsymbol{\sigma}_{t_j}\right)$, for states $\boldsymbol{x}_m^{(k,\,t_j)} \in \mathcal{X}_{t_j}$ and actions $\boldsymbol{u}_m^{(k,\,t_j)} \in \mathcal{U}_{t_j}$. For $l_{t_j}(\boldsymbol{\alpha}_{t_j}) = -\frac{1}{n_{t_j}}\sum_{k=1}^{n_{t_j}}\sum_{m=0}^{M_{t_j}-1}\log\left[\pi_{\boldsymbol{\alpha}_{t_j}}^{(t_j)}\left(\boldsymbol{u}_m^{(k,\,t_j)}|\boldsymbol{x}_m^{(k,\,t_j)}\right)\right]$, the gradient $\nabla_{\boldsymbol{\alpha}_{t_j}} l_{t_j}(\boldsymbol{\alpha}_{t_j})\big|_{\hat{\boldsymbol{\theta}}_r}$ satisfies $\left\|\nabla_{\boldsymbol{\alpha}_{t_j}} l_{t_j}(\boldsymbol{\alpha}_{t_j})\big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2 \leq \frac{M_{t_j}}{\sigma_{t_j}^2}\left(u_{\max} + \max_{t_k\in\mathcal{I}_{r-1}}\left\{\left|\left|\boldsymbol{A}_{t_k}^+\right|\right|_2\left(||\boldsymbol{b}_{t_k}||_2 + \boldsymbol{c}_{\max}\right)\right\}\boldsymbol{\Phi}_{\max}\right)\boldsymbol{\Phi}_{\max}$ for all trajectories and all tasks, with $u_{\max} = \max_{k,m}\left\{\left|\boldsymbol{u}_m^{(k,\,t_j)}\right|\right\}$ and $\boldsymbol{\Phi}_{\max} = \max_{k,m}\left\{\left|\left|\boldsymbol{\Phi}\left(\boldsymbol{x}_m^{(k,\,t_j)}\right)\right|\right|_2\right\}$.*

5.2. Bounding Linearized Losses

As discussed previously, we linearize the loss of task t_r around the constraint solution of the previous round $\hat{\boldsymbol{\theta}}_r$. To acquire the regret bounds in Theorem 1, the next step is to bound the dual norm, $\left\|\hat{\boldsymbol{f}}_{t_r}\big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2^\star = \left\|\hat{\boldsymbol{f}}_{t_r}\big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2$ of Eq. (9). It can be easily seen

$$\left\|\hat{\boldsymbol{f}}_{t_r}\big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2 \leq \underbrace{\left|l_{t_r}(\boldsymbol{\theta})\big|_{\hat{\boldsymbol{\theta}}_r}\right|}_{\text{constant}} + \underbrace{\left\|\nabla_{\boldsymbol{\theta}} l_{t_r}(\boldsymbol{\theta})\big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2}_{\text{Lemma 2}} + \left\|\nabla_{\boldsymbol{\theta}} l_{t_r}(\boldsymbol{\theta})\big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2 \times \underbrace{\left\|\hat{\boldsymbol{\theta}}_r\right\|_2}_{\text{Lemma 3}} . \tag{11}$$

[3]Please note that derivations for other forms of log-concave policy distributions could be derived in similar manner. In this work, we focus on Gaussian policies since they cover a broad spectrum of real-world applications.

Since $\left|l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\right|$ can be bounded by $\boldsymbol{\delta}_{l_{t_r}}$ (see Sect. 2), the next step is to bound $\left\|\nabla_{\boldsymbol{\theta}} l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2$, and $\|\hat{\boldsymbol{\theta}}_r\|_2$.

Lemma 2. *The norm of the gradient of the loss function evaluated at $\hat{\boldsymbol{\theta}}_r$ satisfies*

$$\left\|\nabla_{\boldsymbol{\theta}} l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2^2 \leq \left\|\nabla_{\boldsymbol{\alpha}_{t_r}} l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2^2 \Bigg(q \times d \left(2d/p^2 \max_{t_k \in \mathcal{I}_{r-1}} \left\{\left\|\boldsymbol{A}_{t_k}^{\dagger}\right\|_2^2 \left(\|\boldsymbol{b}_{t_k}\|_2^2 + \boldsymbol{c}_{max}^2\right)\right\} + 1\right)\Bigg) .$$

To finalize the bound of $\left\|\hat{\boldsymbol{f}}_{t_r}\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2$ as needed for deriving the regret, we must derive an upper-bound for $\|\hat{\boldsymbol{\theta}}_r\|_2$:

Lemma 3. *The L_2 norm of the constraint solution at round $r-1$, $\|\hat{\boldsymbol{\theta}}_r\|_2^2$ is bounded by*

$$\|\hat{\boldsymbol{\theta}}_r\|_2^2 \leq q \times d \Bigg[1 + |\mathcal{I}_{r-1}| \frac{1}{p^2} \max_{t_k \in \mathcal{I}_{r-1}} \left\{\left\|\boldsymbol{A}_{t_k}^{\dagger}\right\|_2^2 \left(\|\boldsymbol{b}_{t_k}\|_2 + \boldsymbol{c}_{max}\right)^2\right\}\Bigg] ,$$

where $|\mathcal{I}_{r-1}|$ is the number of unique tasks observed so far.

Given the previous two lemmas, we can prove the bound for $\left\|\hat{\boldsymbol{f}}_{t_r}\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2$:

Lemma 4. *The L_2 norm of the linearizing term of $l_{t_r}(\boldsymbol{\theta})$ around $\hat{\boldsymbol{\theta}}_r$, $\left\|\hat{\boldsymbol{f}}_{t_r}\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2$, is bounded by*

$$\left\|\hat{\boldsymbol{f}}_{t_r}\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2 \leq \left\|\nabla_{\boldsymbol{\theta}} l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2 \left(1 + \|\hat{\boldsymbol{\theta}}_r\|_2\right) + \left|l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\right| \quad (12)$$
$$\leq \boldsymbol{\gamma}_1(r)\left(1 + \boldsymbol{\gamma}_2(r)\right) + \boldsymbol{\delta}_{l_{t_r}} ,$$

where $\boldsymbol{\delta}_{l_{t_r}}$ is the constant upper-bound on $\left|l_{t_r}(\boldsymbol{\theta})\Big|_{\hat{\boldsymbol{\theta}}_r}\right|$, and

$$\boldsymbol{\gamma}_1(r) = \frac{1}{n_{t_j}\sigma_{t_j}^2}\Bigg[\Bigg(u_{max} + \max_{t_k \in \mathcal{I}_{r-1}} \left\{\left\|\boldsymbol{A}_{t_k}^{+}\right\|_2 \left(\|\boldsymbol{b}_{t_k}\|_2 + \boldsymbol{c}_{max}\right)\right\} \boldsymbol{\Phi}_{max}\Bigg) \boldsymbol{\Phi}_{max}\Bigg] \times \left(\frac{d}{p}\sqrt{2q}\sqrt{\max_{t_k \in \mathcal{I}_{r-1}} \left\{\|\boldsymbol{A}_{t_k}^{\dagger}\|_2^2 \left(\|\boldsymbol{b}_{t_k}\|_2^2 + \boldsymbol{c}_{max}^2\right)\right\}} + \sqrt{qd}\right)$$

$$\boldsymbol{\gamma}_2(r) \leq \sqrt{q \times d} + \sqrt{|\mathcal{I}_{r-1}|}\sqrt{1 + \frac{1}{p^2} \max_{t_k \in \mathcal{I}_{r-1}} \left\{\left\|\boldsymbol{A}_{t_k}^{\dagger}\right\|_2^2 \left(\|\boldsymbol{b}_{t_k}\|_2 + \boldsymbol{c}_{max}\right)^2\right\}} .$$

5.3. Completing the Proof of Sublinear Regret

Given the lemmas in the previous section, we now can derive the sublinear regret bound given in Theorem 1. Using results developed by Abbasi-Yadkori et al. (2013), it is easy to see that

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\Omega}_0\left(\tilde{\boldsymbol{\theta}}_j\right) - \nabla_{\boldsymbol{\theta}} \boldsymbol{\Omega}_0\left(\tilde{\boldsymbol{\theta}}_{j+1}\right) = \eta_{t_j} \hat{\boldsymbol{f}}_{t_j}\Big|_{\hat{\boldsymbol{\theta}}_j} .$$

From the convexity of the regularizer, we obtain:

$$\boldsymbol{\Omega}_0\left(\hat{\boldsymbol{\theta}}_j\right) \geq \boldsymbol{\Omega}_0\left(\hat{\boldsymbol{\theta}}_{j+1}\right) + \left\langle \nabla_{\boldsymbol{\theta}} \boldsymbol{\Omega}_0\left(\hat{\boldsymbol{\theta}}_{j+1}\right), \hat{\boldsymbol{\theta}}_j - \hat{\boldsymbol{\theta}}_{j+1}\right\rangle + \frac{1}{2}\left\|\hat{\boldsymbol{\theta}}_j - \hat{\boldsymbol{\theta}}_{j+1}\right\|_2^2 .$$

We have:

$$\left\|\hat{\boldsymbol{\theta}}_j - \hat{\boldsymbol{\theta}}_{j+1}\right\|_2 \leq \eta_{t_j} \left\|\hat{\boldsymbol{f}}_{t_j}\Big|_{\hat{\boldsymbol{\theta}}_j}\right\|_2 .$$

Therefore, for any $\boldsymbol{u} \in \mathcal{K}$

$$\sum_{j=1}^{r} \eta_{t_j}\left(l_{t_j}\left(\hat{\boldsymbol{\theta}}_j\right) - l_{t_j}(\boldsymbol{u})\right) \leq \sum_{j=1}^{r} \eta_{t_j} \left\|\hat{\boldsymbol{f}}_{t_j}\Big|_{\hat{\boldsymbol{\theta}}_j}\right\|_2^2 + \boldsymbol{\Omega}_0(\boldsymbol{u}) - \boldsymbol{\Omega}_0(\hat{\boldsymbol{\theta}}_1) .$$

Assuming that $\forall t_j\ \eta_{t_j} = \eta$, we can derive:

$$\sum_{j=1}^{r}\left(l_{t_j}\left(\hat{\boldsymbol{\theta}}_j\right) - l_{t_j}(\boldsymbol{u})\right) \leq \eta \sum_{j=1}^{r} \left\|\hat{\boldsymbol{f}}_{t_j}\Big|_{\hat{\boldsymbol{\theta}}_j}\right\|_2^2 + {}^1\!/\!_\eta \left(\boldsymbol{\Omega}_0(\boldsymbol{u}) - \boldsymbol{\Omega}_0(\hat{\boldsymbol{\theta}}_1)\right) .$$

The following lemma finalizes the proof of Theorem 1:

Lemma 5. *After R rounds with $\forall t_j\ \eta_{t_j} = \eta = \frac{1}{\sqrt{R}}$, for any $\boldsymbol{u} \in \mathcal{K}$ we have that $\sum_{j=1}^{R} l_{t_j}(\hat{\boldsymbol{\theta}}_j) - l_{t_j}(\boldsymbol{u}) \leq \mathcal{O}\left(\sqrt{R}\right)$.*

Proof. From Eq. (12), it follows that

$$\left\|\hat{\boldsymbol{f}}_{t_j}\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2^2 \leq \boldsymbol{\gamma}_3(R) + 4\boldsymbol{\gamma}_1^2(R)\boldsymbol{\gamma}_2^2(R) \leq \boldsymbol{\gamma}_3(R) + 8\frac{d}{p^2}\boldsymbol{\gamma}_1^2(R) qd \Bigg(1 + |\mathcal{I}_{R-1}| \times \max_{t_k \in \mathcal{I}_{R-1}} \left\{\|\boldsymbol{A}_{t_k}^{\dagger}\|_2 \left(\|\boldsymbol{b}_{t_k}\|_2 + \boldsymbol{c}_{\max}\right)^2\right\}\Bigg)$$

with $\boldsymbol{\gamma}_3(R) = 4\boldsymbol{\gamma}_1^2(R) + 2\max_{t_j \in \mathcal{I}_{R-1}} \boldsymbol{\delta}_{t_j}^2$. Since $|\mathcal{I}_{R-1}| \leq |\mathcal{T}|$, we have that $\left\|\hat{\boldsymbol{f}}_{t_j}\Big|_{\hat{\boldsymbol{\theta}}_r}\right\|_2^2 \leq \boldsymbol{\gamma}_5(R)|\mathcal{T}|$ with $\boldsymbol{\gamma}_5 = 8^d/_{p^2} q\boldsymbol{\gamma}_1^2(R) \max_{t_k \in \mathcal{I}_{R-1}} \left\{\|\boldsymbol{A}_{t_k}^{\dagger}\|_2^2 \left(\|\boldsymbol{b}_{t_k}\|_2 + \boldsymbol{c}_{\max}\right)^2\right\}$.

Given that $\boldsymbol{\Omega}_0(\boldsymbol{u}) \leq qd + \boldsymbol{\gamma}_5(R)|\mathcal{T}|$, with $\boldsymbol{\gamma}_5(R)$ being a constant, we have:

$$\sum_{j=1}^{r}\left(l_{t_j}\left(\hat{\boldsymbol{\theta}}_j\right) - l_{t_j}(\boldsymbol{u})\right) \leq \eta \sum_{j=1}^{r} \boldsymbol{\gamma}_5(R)|\mathcal{T}| + \frac{1}{\eta}\left(qd + \boldsymbol{\gamma}_5(R)|\mathcal{T}| - \boldsymbol{\Omega}_0(\hat{\boldsymbol{\theta}}_1)\right) .$$

Initializing $\boldsymbol{L}$ and $\boldsymbol{S}$: We initialize $\boldsymbol{L}\Big|_{\hat{\boldsymbol{\theta}}_1} = \text{diag}_{\boldsymbol{k}}(\zeta)$, with $p \leq \zeta^2 \leq q$ and $\boldsymbol{S}\Big|_{\hat{\boldsymbol{\theta}}_1} = \boldsymbol{0}_{k \times |\mathcal{T}|}$ to ensure the invertibility

of $\boldsymbol{L}$ and that the constraints are met. This leads to

$$\sum_{j=1}^{r}\left(l_{t_j}\left(\hat{\boldsymbol{\theta}}_j\right)-l_{t_j}(\boldsymbol{u})\right) \leq \eta\sum_{j=1}^{r}\gamma_5(R)|\mathcal{T}| + {}^1/_\eta\left(qd+\gamma_5(R)|\mathcal{T}|-\mu_2 k\zeta\right) .$$

Choosing $\forall t_j\ \eta_{t_j}=\eta={}^1/\sqrt{R}$, we acquire sublinear regret, finalizing the statement of Theorem 1:

$$\begin{aligned}\sum_{j=1}^{r}\left(l_{t_j}\left(\hat{\boldsymbol{\theta}}_j\right)-l_{t_j}(\boldsymbol{u})\right) &\leq {}^1/\sqrt{R}\gamma_5(R)|\mathcal{T}|R \\ &\quad + \sqrt{R}\left(qd+\gamma_5(R)|\mathcal{T}|-\mu_2 k\zeta\right) \\ &\leq \sqrt{R}\Big(\gamma_5(R)|\mathcal{T}|+qd\gamma_5(R)|\mathcal{T}|-\mu_2 k\zeta\Big) \\ &\leq \mathcal{O}\left(\sqrt{R}\right) . \qquad \square\end{aligned}$$

6. Experimental Validation

To validate the empirical performance of our method, we applied our safe online PG algorithm to learn multiple consecutive control tasks on three dynamical systems (Figure 1). To generate multiple tasks, we varied the parameterization of each system, yielding a set of control tasks from each domain with varying dynamics. The optimal control policies for these systems vary widely with only minor changes in the system parameters, providing substantial diversity among the tasks within a single domain.

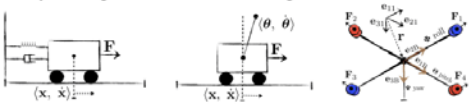


Figure 1. Dynamical systems used in the experiments: *a)* simple mass system (left), *b)* cart-pole (middle), and *c)* quadrotor unmanned aerial vehicle (right).

Simple Mass Spring Damper: The simple mass (SM) system is characterized by three parameters: the spring constant k in N/m, the damping constant d in Ns/m and the mass m in kg. The system's state is given by the position $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ of the mass, which varies according to a linear force $\boldsymbol{F}$. The goal is to train a policy for controlling the mass in a specific state $\boldsymbol{g}_{\text{ref}}=\langle \boldsymbol{x}_{\text{ref}}, \dot{\boldsymbol{x}}_{\text{ref}}\rangle$.

Cart Pole: The cart-pole (CP) has been used extensively as a benchmark for evaluating RL methods (Busoniu et al., 2010). CP dynamics are characterized by the cart's mass m_c in kg, the pole's mass m_p in kg, the pole's length in meters, and a damping parameter d in Ns/m. The state is given by the cart's position $\boldsymbol{x}$ and velocity $\dot{\boldsymbol{x}}$, as well as the pole's angle $\boldsymbol{\theta}$ and angular velocity $\dot{\boldsymbol{\theta}}$. The goal is to train a policy that controls the pole in an upright position.

6.1. Experimental Protocol

We generated 10 tasks for each domain by varying the system parameters to ensure a variety of tasks with diverse optimal policies, including those with highly chaotic dynamics that are difficult to control. We ran each experiment for a total of R rounds, varying from 150 for the simple mass to $10,000$ for the quadrotor to train $\boldsymbol{L}$ and $\boldsymbol{S}$, as well as for updating the PG-ELLA and PG models. At each round j, the learner observed a task t_j through 50 trajectories of 150 steps and updated $\boldsymbol{L}$ and $\boldsymbol{s}_{t_j}$. The dimensionality k of the latent space was chosen independently for each domain via cross-validation over 3 tasks, and the learning step size for each task domain was determined by a line search after gathering 10 trajectories of length 150. We used eNAC, a standard PG algorithm, as the base learner.

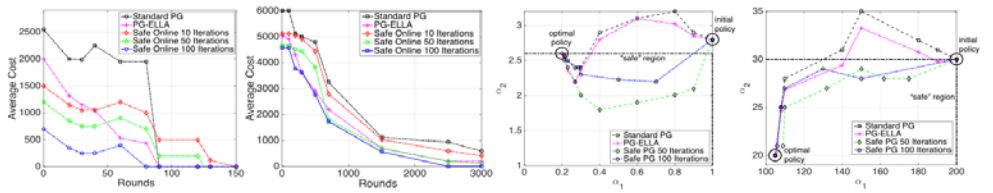
We compared our approach to both standard PG (i.e., eNAC) and PG-ELLA (Bou Ammar et al., 2014), examining both the constrained and unconstrained variants of our algorithm. We also varied the number of iterations in our alternating optimization from 10 to 100 to evaluate the effect of these inner iterations on the performance, as shown in Figures 2 and 3. For the two MTL algorithms (our approach and PG-ELLA), the policy parameters for each task t_j were initialized using the learned basis (i.e., $\boldsymbol{\alpha}_{t_j}=\boldsymbol{L}\boldsymbol{s}_{t_j}$). We configured PG-ELLA as described by Bou Ammar et al. (2014), ensuring a fair comparison. For the standard PG learner, we provided additional trajectories in order to ensure a fair comparison, as described below.

For the experiments with policy constraints, we generated a set of constraints $(\boldsymbol{A}_t, \boldsymbol{b}_t)$ for each task that restricted the policy parameters to pre-specified "safe" regions, as shown in Figures 2(c) and 2(d). We also tested different values for the constraints on $\boldsymbol{L}$, varying p and q between 0.1 to 10; our approach showed robustness against this broad range, yielding similar average cost performance.

6.2. Results on Benchmark Systems

Figure 2 reports our results on the benchmark simple mass and cart-pole systems. Figures 2(a) and 2(b) depicts the performance of the learned policy in a lifelong learning setting over consecutive unconstrained tasks, averaged over all 10 systems over 100 different initial conditions. These results demonstrate that our approach is capable of outperforming both standard PG (which was provided with 50 *additional* trajectories each iteration to ensure a more fair comparison) and PG-ELLA, both in terms of initial performance and learning speed. These figures also show that the performance of our method increases as it is given more alternating iterations per-round for fitting $\boldsymbol{L}$ and $\boldsymbol{S}$.

We evaluated the ability of these methods to respect safety constraints, as shown in Figures 2(c) and 2(d). The thicker black lines in each figure depict the allowable "safe" region of the policy space. To enable online learning per-task, the same task t_j was observed on each round and the shared basis $\boldsymbol{L}$ and coefficients $\boldsymbol{s}_{t_j}$ were updated using alternating optimization. We then plotted the change in the policy pa-



(a) Simple Mass (b) Cart Pole (c) Trajectory Simple Mass (d) Trajectory Cart Pole

Figure 2. Results on benchmark simple mass and cart-pole systems. Figures (a) and (b) depict performance in lifelong learning scenarios over consecutive unconstrained tasks, showing that our approach outperforms standard PG and PG-ELLA. Figures (c) and (d) examine the ability of these method to abide by safety constraints on sample constrained tasks, depicting two dimensions of the policy space (α_1 vs α_2) and demonstrating that our approach abides by the constraints (the dashed black region).

rameter vectors per iterations (i.e., $\boldsymbol{\alpha}_{t_j} = \boldsymbol{L}\boldsymbol{s}_{t_j}$) for each method, demonstrating that our approach abides by the safety constraints, while standard PG and PG-ELLA can violate them (since they only solve an unconstrained optimization problem). In addition, these figures show that increasing the number of alternating iterations in our method causes it to take a more direct path to the optimal solution.

6.3. Application to Quadrotor Control

We also applied our approach to the more challenging domain of quadrotor control. The dynamics of the quadrotor system (Figure 1) are influenced by inertial constants around $\boldsymbol{e}_{1,B}$, $\boldsymbol{e}_{2,B}$, and $\boldsymbol{e}_{3,B}$, thrust factors influencing how the rotor's speed affects the overall variation of the system's state, and the lengths of the rods supporting the rotors. Although the overall state of the system can be described by a 12-dimensional vector, we focus on stability and so consider only six of these state-variables. The quadrotor system has a high-dimensional action space, where the goal is to control the four rotational velocities $\{w_i\}_{i=1}^4$ of the rotors to stabilize the system. To ensure realistic dynamics, we used the simulated model described by (Bouabdallah, 2007; Voos & Bou Ammar, 2010), which has been verified and used in the control of physical quadrotors.

We generated 10 different quadrotor systems by varying the inertia around the x, y and z-axes. We used a linear quadratic regulator, as described by Bouabdallah (2007), to initialize the policies in both the learning and testing phases. We followed a similar experimental procedure to that discussed above to update the models.

Figure 3 shows the performance of the unconstrained solution as compared to standard PG and PG-ELLA. Again, our approach clearly outperforms standard PG and PG-ELLA in both the initial performance and learning speed. We also evaluated constrained tasks in a similar manner, again showing that our approach is capable of respecting constraints. Since the policy space is higher dimensional, we cannot visualize it as well as the benchmark systems, and so instead report the number of iterations it takes our approach

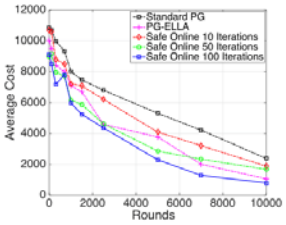


Figure 3. Performance on quadrotor control.

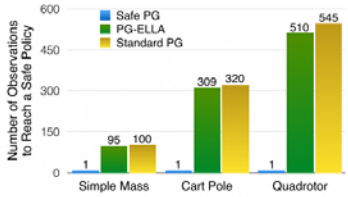


Figure 4. Average number of task observations before acquiring policy parameters that abide by the constraints, showing that our approach immediately projects policies to safe regions.

to project the policy into the safe region. Figure 4 shows that our approach requires only one observation of the task to acquire safe policies, which is substantially lower then standard PG or PG-ELLA (e.g., which require 545 and 510 observations, respectively, in the quadrotor scenario).

7. Conclusion

We described the first lifelong PG learner that provides sublinear regret $\mathcal{O}(\sqrt{R})$ with R total rounds. In addition, our approach supports safety constraints on the learned policy, which are essential for robust learning in real applications. Our framework formalizes lifelong learning as online MTL with limited resources, and enables safe transfer by sharing policy parameters through a latent knowledge base that is efficiently updated over time.

Acknowledgements

This research was supported by ONR grant #N00014-11-1-0139 and AFRL grant #FA8750-14-1-0069. We thank Ali Jadbabaie for assistance with the optimization solution, and the anonymous reviewers for their helpful feedback.

References

Yasin Abbasi-Yadkori, Peter Bartlett, Varun Kanade, Yevgeny Seldin, & Csaba Szepesvári. Online learning in Markov decision processes with adversarially chosen transition probability distributions. *Advances in Neural Information Processing Systems* 26, 2013.

Haitham Bou Ammar, Karl Tuyls, Matthew E. Taylor, Kurt Driessen, & Gerhard Weiss. Reinforcement learning transfer via sparse coding. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems* (AAMAS), 2012.

Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, & Matthew Taylor. Online multi-task learning for policy gradient methods. In *Proceedings of the 31st International Conference on Machine Learning* (ICML), 2014.

Samir Bouabdallah. *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD Thesis, École polytechnique fédérale de Lausanne, 2007.

Stephen Boyd & Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, 2004.

Lucian Busoniu, Robert Babuska, Bart De Schutter, & Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, FL, 2010.

Eliseo Ferrante, Alessandro Lazaric, & Marcello Restelli. Transfer of task representation in reinforcement learning using policy-based proto-value functions. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS), 2008.

Mohammad Gheshlaghi Azar, Alessandro Lazaric, & Emma Brunskill. Sequential transfer in multi-armed bandit with finite set of models. *Advances in Neural Information Processing Systems* 26, 2013.

Roger A. Horn & Roy Mathias. Cauchy-Schwarz inequalities associated with positive semidefinite matrices. *Linear Algebra and its Applications* 142:63–82, 1990.

Jens Kober & Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1–2):171–203, 2011.

Abhishek Kumar & Hal Daumé III. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning* (ICML), 2012.

Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In M. Wiering & M. van Otterlo, editors, *Reinforcement Learning: State of the Art*. Springer, 2011.

Jan Peters & Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 2008a.

Jan Peters & Stefan Schaal. Natural Actor-Critic. *Neurocomputing* 71, 2008b.

Paul Ruvolo & Eric Eaton. ELLA: An Efficient Lifelong Learning Algorithm. In *Proceedings of the 30th International Conference on Machine Learning* (ICML), 2013.

Richard S. Sutton & Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

Richard S. Sutton, David Mcallester, Satinder Singh, & Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems* 12, 2000.

Matthew E. Taylor & Peter Stone. Transfer learning for reinforcement learning domains: a survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

Sebastian Thrun & Joseph O'Sullivan. Discovering structure in multiple learning tasks: the TC algorithm. In *Proceedings of the 13th International Conference on Machine Learning* (ICML), 1996a.

Sebastian Thrun & Joseph O'Sullivan. Learning more from less data: experiments in lifelong learning. *Seminar Digest*, 1996b.

Holger Voos & Haitham Bou Ammar. Nonlinear tracking and landing controller for quadrotor aerial robots. In *Proceedings of the IEEE Multi-Conference on Systems and Control*, 2010.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3–4):229–256, 1992.

Aaron Wilson, Alan Fern, Soumya Ray, & Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning* (ICML), 2007.

Jian Zhang, Zoubin Ghahramani, & Yiming Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

Appendix D: Bou Aamar et al., 2015c

Autonomous Cross-Domain Knowledge Transfer in Lifelong Policy Gradient Reinforcement Learning

Haitham Bou Ammar
Univ. of Pennsylvania
haithamb@seas.upenn.edu

Eric Eaton
Univ. of Pennsylvania
eeaton@cis.upenn.edu

José Marcio Luna
Univ. of Pennsylvania
joseluna@seas.upenn.edu

Paul Ruvolo
Olin College of Engineering
paul.ruvolo@olin.edu

Abstract

Online multi-task learning is an important capability for lifelong learning agents, enabling them to acquire models for diverse tasks over time and rapidly learn new tasks by building upon prior experience. However, recent progress toward lifelong reinforcement learning (RL) has been limited to learning from within a single task domain. For truly versatile lifelong learning, the agent must be able to autonomously transfer knowledge between different task domains. A few methods for cross-domain transfer have been developed, but these methods are computationally inefficient for scenarios where the agent must learn tasks consecutively.

In this paper, we develop the first cross-domain lifelong RL framework. Our approach efficiently optimizes a shared repository of transferable knowledge and learns projection matrices that specialize that knowledge to different task domains. We provide rigorous theoretical guarantees on the stability of this approach, and empirically evaluate its performance on diverse dynamical systems. Our results show that the proposed method can learn effectively from interleaved task domains and rapidly acquire high performance in new domains.

1 Introduction

Reinforcement learning (RL) provides the ability to solve high-dimensional control problems when detailed knowledge of the system is not available *a priori*. Applications with these characteristics are ubiquitous in a variety of domains, from robotic control [Busoniu *et al.*, 2008; Smart & Kaelbling, 2002] to stock trading [Dempster & Leemans, 2006]. However, in many cases, RL methods require numerous lengthy interactions with the dynamical system in order to learn an acceptable controller. Unfortunately, the cost of acquiring these interactions is often prohibitively expensive (in terms of time, expense, physical wear on the robot, etc.).

This issue of obtaining adequate experience only worsens when *multiple* control problems must be solved. This need arises in two main cases: (1) when a single agent must learn to solve multiple tasks (e.g., a reconfigurable robot), and (2) when different robots must each learn to solve a single control problem. *Transfer learning* [Taylor & Stone, 2009] and *multi-task learning* (MTL) methods [Li *et al.*, 2009; Lazaric & Ghavamzadeh, 2010] have been developed to reduce the amount of experience needed for individual tasks by allowing the agent to reuse knowledge from other tasks.

In both transfer learning and MTL, the difficulty of transferring knowledge between tasks increases with the diversity of the tasks. In the extreme case, the underlying systems (and their action and state representations) are entirely different, making direct knowledge reuse impossible. Several approaches to this *cross-domain transfer* problem have been developed, but these methods require an inter-task mapping of state/action spaces that is either hand-coded [Taylor *et al.*, 2007] or learned in a computationally inefficient manner that does not scale to more than a few task domains (see Section 3). However, the problem of cross-domain transfer has not yet been studied in lifelong learning settings [Thrun & O'Sullivan, 1996; Ruvolo & Eaton, 2013], in which the agent must learn multiple tasks consecutively with the goal of optimizing performance across all previously learned tasks.

We address this problem by developing the first algorithm for lifelong RL that supports efficient and autonomous cross-domain transfer between multiple consecutive tasks from different domains. Specifically, our approach provides the following advantages over existing approaches: (1) it improves current cross-domain transfer learning methods by optimizing performance across all tasks, (2) it learns multi-task cross-domain mappings autonomously, and (3) it can share knowledge between a multitude of tasks from diverse domains in a computationally efficient manner. To enable effective cross-domain lifelong learning, our approach learns a repository of shared knowledge along with projection matrices that specialize this shared knowledge to each task domain. We provide theoretical guarantees on convergence that show this approach becomes increasingly stable as the number of domains or tasks grows large, and demonstrate the empirical effectiveness of cross-domain lifelong learning between streams of interleaved tasks from diverse dynamical systems, including bicycles and helicopters.

2 Background on Policy Gradient RL

In a reinforcement learning (RL) problem, an agent must decide how to sequentially select actions to maximize its ex-

pected return. In contrast to classic stochastic optimal control methods [Bertsekas, 1995], RL approaches do not require detailed prior knowledge of the system dynamics or goal; instead these approaches learn optimal control policies through interaction with the system itself. RL problems are typically formalized as a Markov decision process (MDP) $\langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the (potentially infinite) set of states, $\mathcal{A}$ is the set of actions that the agent may execute, $P : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is a state transition probability function describing the task dynamics, $R : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ is the reward function measuring the performance of the agent, and $\gamma \in [0, 1)$ is the reward discount factor. At each time step m, the agent is in state $\boldsymbol{x}_m \in \mathcal{X}$ and must choose an action $\boldsymbol{a}_m \in \mathcal{A}$, transitioning it to a new state $\boldsymbol{x}_{m+1} \sim P(\boldsymbol{x}_{m+1} \mid \boldsymbol{x}_m, \boldsymbol{a}_m)$ and yielding a reward $r_{m+1} = R(\boldsymbol{x}_m, \boldsymbol{a}_m, \boldsymbol{x}_{m+1})$. The sequence of state-action pairs forms a trajectory $\boldsymbol{\tau} = [\boldsymbol{x}_{1:M}, \boldsymbol{a}_{1:M}]$ over a (possibly infinite) horizon M. A policy $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ specifies a conditional probability distribution over actions given the current state. The RL agent's goal is to find a policy $\pi^\star$ that maximizes the expected per-time-step reward.

Policy gradient methods [Peters *et al.*, 2005; Sutton *et al.*, 1999] represent the agent's policy π as a function defined over a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ of control parameters. With this parameterized policy, we can compute the optimal parameters $\boldsymbol{\theta}^\star$ that maximize the expected average reward:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathrm{E}\left[\frac{1}{M}\sum_{m=1}^{M} r_{m+1}\right] = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\mathfrak{R}(\boldsymbol{\tau})d\boldsymbol{\tau} \ , \qquad (1)$$

where $\mathbb{T}$ is the set of all possible trajectories, $\mathfrak{R}(\boldsymbol{\tau}) = \frac{1}{M}\sum_{m=1}^{M} r_{m+1}$ is the reward of trajectory $\boldsymbol{\tau}$, and $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = P_0(\mathbf{x}_1)\prod_{m=1}^{M} P(\mathbf{x}_{m+1} \mid \mathbf{x}_m, \boldsymbol{a}_m)\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_m \mid \mathbf{x}_m)$ is the probability of $\boldsymbol{\tau}$ with initial state distribution $P_0 : \mathcal{X} \to [0, 1]$.

To maximize $\mathcal{J}(\cdot)$, most policy gradient algorithms (e.g., episodic REINFORCE [Williams, 1992], PoWER [Rückstieß *et al.*, 2008], and Natural Actor Critic [Peters & Schaal, 2008]) employ standard supervised function approximation to learn $\boldsymbol{\theta}$ by maximizing a lower bound on $\mathcal{J}(\boldsymbol{\theta})$. To maximize this lower bound, these methods generate trajectories using the current policy $\pi_{\boldsymbol{\theta}}$, and then compare the result with a new policy $\pi_{\tilde{\boldsymbol{\theta}}}$. Kober & Peters [2011] describe how this lower bound on the expected return can be attained using Jensen's inequality and the concavity of the logarithm:

$$\begin{aligned}
\log \mathcal{J}\left(\tilde{\boldsymbol{\theta}}\right) &= \log \int_{\mathbb{T}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}\, p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})\, \mathfrak{R}(\boldsymbol{\tau})\, \mathrm{d}\boldsymbol{\tau} \\
&\geq \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\, \mathfrak{R}(\boldsymbol{\tau}) \log \frac{p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}\, \mathrm{d}\boldsymbol{\tau} + \text{constant} \\
&\propto -\mathfrak{D}_{\mathrm{KL}}\big(p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\, \mathfrak{R}(\boldsymbol{\tau}) \,||\, p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})\big) = \mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}\right) \ ,
\end{aligned}$$

where $\mathfrak{D}_{\mathrm{KL}}\big(p(\boldsymbol{\tau}) \,||\, q(\boldsymbol{\tau})\big) = \int_{\mathbb{T}} p(\boldsymbol{\tau}) \log \frac{p(\boldsymbol{\tau})}{q(\boldsymbol{\tau})}\, \mathrm{d}\boldsymbol{\tau}$. Consequently, this process is equivalent to minimizing the KL divergence $\mathfrak{D}_{\mathrm{KL}}$ between $\pi_{\boldsymbol{\theta}}$'s reward-weighted trajectory distribution and the trajectory distribution $p_{\tilde{\boldsymbol{\theta}}}$ of $\pi_{\tilde{\boldsymbol{\theta}}}$.

Policy gradient methods have gained attention in the RL community in part due to their successful applications to robotics [Peters *et al.*, 2005]. While such methods have a low computational cost per update, high-dimensional problems require many updates (by acquiring new rollouts) to achieve good performance. Transfer learning and multi-task learning can reduce these data requirements and accelerate learning.

3 Related Work on RL Knowledge Transfer

Knowledge transfer between tasks has been explored in the context of transfer learning and multi-task learning. In all cases, each task t is described by an MDP $\mathcal{Z}^{(t)} = \langle \mathcal{X}^{(t)}, \mathcal{A}^{(t)}, P^{(t)}, R^{(t)}, \gamma^{(t)} \rangle$.

Transfer learning aims to improve the learning time and/or behavior of the agent on a new target task by transferring knowledge learned from one or more previous source tasks [Taylor & Stone, 2009]. However, transfer learning methods only focus on optimizing performance on the target task, and therefore are not ideal for agents that revisit earlier tasks. In contrast, *multi-task learning* (MTL) methods [Li *et al.*, 2009; Lazaric & Ghavamzadeh, 2010] optimize performance over all tasks, often by training task models simultaneously while sharing knowledge between tasks, but are computationally expensive in lifelong learning scenarios where the agent must learn tasks consecutively over time [Ruvolo & Eaton, 2013; Thrun & O'Sullivan, 1996], as we explore in this paper.

One exception is PG-ELLA [Bou Ammar *et al.*, 2014], a recent lifelong policy gradient RL algorithm that can efficiently learn multiple tasks consecutively while sharing knowledge between task policies to accelerate learning. In fact, PG-ELLA can be viewed as a special case of the algorithm we propose in this paper where learning is limited to only tasks from a single domain. MTL for policy gradients has also been explored by Deisenroth et al. [2014] through customizing a single parameterized controller to individual tasks that differ only in the reward function. Another closely related work is on hierarchical Bayesian MTL [Wilson *et al.*, 2007], which can learn RL tasks consecutively, but unlike our approach, requires discrete states and actions. Snel and Whiteson's [2014] representation learning approach is also related, but assumes all tasks share the same feature and action sets. All of these MTL methods operate on tasks from a single domain, and do not support cross-domain transfer.

To enable transfer between tasks with different state and/or action spaces, transfer learning and MTL methods require an inter-task mapping to translate knowledge between tasks. The earliest work on cross-domain transfer in RL, by Taylor et al., required a hand-coded mapping [2007] or a computationally expensive exploration of all permitted mappings [2008]. More recently, unsupervised techniques have been developed to autonomously learn inter-task mappings [Bou Ammar *et al.* 2012; 2013]. While these approaches enable autonomous cross-domain transfer, they only learn pairwise mappings between tasks and are computationally expensive, making them inapplicable for transfer among numerous tasks from different domains. In contrast to these methods, our approach provides a computationally efficient mechanism for transfer between multiple task domains, enabling cross-domain lifelong

RL. Cross-domain MTL has also been explored in a limited fashion in supervised settings [Han *et al.*, 2012].

4 Problem Definition

Previous work on policy gradients has focused on either single-task learning or MTL within a single task domain (i.e., all tasks share a common state and action space, but may differ in other aspects). We focus on the problem of learning multiple tasks consecutively, where the tasks may be drawn from different task domains. Specifically, the agent must learn a series of RL tasks $\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(T_{\max})}$ over its lifetime, where each task $\mathcal{Z}^{(t)}$ is an MDP and the tasks may have different state and/or action spaces. We can partition the series of RL tasks into task groups $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(G_{\max})}$, such that all tasks within a particular group $\mathcal{G}^{(g)}$ (i.e., a set of tasks) share a common state and action space, and are generated by varying latent parameters [Konidaris & Doshi-Velez, 2014]. In our experiments, each task group corresponds to a task domain—a class of dynamical systems, such as helicopters, cart-poles, etc., each of which contains multiple tasks corresponding to multiple physical helicopters or cart-poles. However, this framework can easily generalize so that a particular group $\mathcal{G}^{(g)}$ could represent a subset of tasks from a domain, similar to task clustering frameworks [Kang *et al.*, 2011].

The agent must learn the tasks consecutively, acquiring multiple trajectories within each task before moving to the next. The tasks may be interleaved, offering the agent the opportunity to revisit earlier tasks (or task domains) for further experience, but the agent has no control over the task order. We assume that *a priori* the agent does not know the total number of tasks $T_{\max}$, their distribution, the task order, or the total number of task groups $G_{\max}$. The agent also has no prior knowledge about the inter-task mappings between tasks, and so it must also learn how to transfer knowledge between task domains in order to optimize overall performance.

The agent's goal is to learn a set of *optimal* policies $\mathbf{\Pi}^\star = \left\{\pi^\star_{\boldsymbol{\theta}^{(1)}}, \ldots, \pi^\star_{\boldsymbol{\theta}^{(T_{\max})}}\right\}$ with corresponding parameters $\mathbf{\Theta}^\star = \{\boldsymbol{\theta}^{(1)\star}, \ldots, \boldsymbol{\theta}^{(T_{\max})\star}\}$. Since tasks belong to different domains, the dimension of the parameter vectors will vary, with $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{d^{(t)}}$ where $d^{(t)}$ is the dimension of the state space $\mathcal{X}^{(t)}$. At any time, the agent may be evaluated on any previous task, and so must strive to optimize its learned policies for all tasks $\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(T)}$, where $T = \sum_{g=1}^{G} |\mathcal{G}^{(g)}|$ denotes the number of tasks seen so far ($1 \leq T \leq T_{\max}$) and G is the number of groups seen so far ($1 \leq G \leq G_{\max}$).

5 Cross-Domain Lifelong RL

This section develops our cross-domain lifelong RL approach. In order to share knowledge between the tasks, we assume that each task's policy parameters $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{d^{(t)}}$ for task $t \in \mathcal{G}^{(g)}$ can be modeled as a sparse linear combination of latent components from knowledge base $B^{(g)} \in \mathbb{R}^{d^{(t)} \times k}$ that is shared among all tasks in the group. Therefore, we have that $\boldsymbol{\theta}^{(t)} = B^{(g)} s^{(t)}$, with sparse task-specific coefficients $s^{(t)} \in \mathbb{R}^k$ for task t. The collection of all task-specific coefficients for tasks in $\mathcal{G}^{(g)}$ is given by $\boldsymbol{S}^{(t \in \mathcal{G}^{(g)})} \in \mathbb{R}^{k \times |\mathcal{G}^{(g)}|}$.

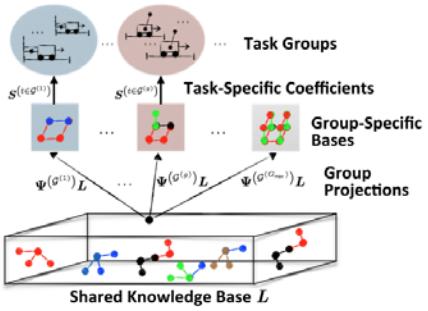


Figure 1: The knowledge framework, showing the shared repository of transferrable knowledge $\boldsymbol{L}$, group projections $\boldsymbol{\Psi}$ that specialize $\boldsymbol{L}$ to each task group, task-specific coefficients over the specialized basis, and task groups.

Effectively, $B^{(g)}$ forms a k-component basis over the policy parameters for all tasks in $\mathcal{G}^{(g)}$, enabling the transfer of knowledge between tasks from this group. The task-specific coefficients $s^{(t)}$ are encouraged to be sparse to ensure that each learned basis component captures a maximal reusable chunk of knowledge. This knowledge framework has been use successfully by a number of other MTL methods [Kumar & Daumé III, 2012; Maurer *et al.*, 2013; Ruvolo & Eaton, 2013] for transfer between tasks within a single task domain.

To support cross-domain transfer, we introduce a repository of knowledge $\boldsymbol{L} \in \mathbb{R}^{d \times k}$ that is shared among all tasks (including between task domains). This matrix $\boldsymbol{L}$ represents a set of latent factors that underly the set of group-specific basis matrices $\{B^{(1)}, \ldots, B^{(G_{\max})}\}$. We introduce a set of group projection matrices $\boldsymbol{\Psi}^{(\mathcal{G}^{(1)})}, \ldots, \boldsymbol{\Psi}^{(\mathcal{G}^{(G_{\max})})}$ that map the shared latent factors $\boldsymbol{L}$ into the basis for each group of tasks. Therefore, we have that $B^{(g)} = \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\boldsymbol{L}$, where the group projection matrix $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \in \mathbb{R}^{d^{(t)} \times d}$. With this construction, we see that each mapping $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}$ creates an intermediate knowledge space (i.e., $B^{(g)}$) that tailors the shared repository $\boldsymbol{L}$ into a basis that is suitable for learning tasks from group $\mathcal{G}^{(g)}$. These group-specific bases are coupled together via the $\boldsymbol{\Psi}$ mappings and the shared knowledge base $\boldsymbol{L}$, facilitating transfer across task domains with different feature spaces. The group mappings $\boldsymbol{\Psi}$'s also serve to help avoid overfitting and ensure compactness of the basis, while maximizing transfer both between tasks within a group and across task groups. This construction is depicted in Figure 1.

5.1 The Cross-Domain MTL Objective

Under this shared knowledge framework, given a task $t \in \mathcal{G}^{(g)}$, its policy parameters $\boldsymbol{\theta}^{(t)} = \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\boldsymbol{L}\boldsymbol{s}^{(t)}$, where $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \in \mathbb{R}^{d_{(t)} \times d}$, $\boldsymbol{L} \in \mathbb{R}^{d \times k}$, $\boldsymbol{s}^{(t)} \in \mathbb{R}^k$, and k is the number of shared latent knowledge components. Therefore, to train optimal policies for all tasks, we must learn the shared knowledge base $\boldsymbol{L}$, the group projections (the $\boldsymbol{\Psi}$'s), and the task-specific coefficients (the $\boldsymbol{s}^{(t)}$'s). We first examine this

problem from a batch MTL standpoint, and then develop an efficient online algorithm for optimizing this MTL objective and enabling lifelong learning in the next section.

Given task groups $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(G)}$, we can represent our objective of learning $T = \sum_{g=1}^{G} |\mathcal{G}^{(g)}|$ stationary policies while maximizing the amount of transfer between task policies as the problem of minimizing:

$$e_{\mathsf{T}}\left(\boldsymbol{L}, \boldsymbol{\Psi}^{(\mathcal{G}^{(1)})}, \ldots, \boldsymbol{\Psi}^{(\mathcal{G}^{(G)})}\right) \tag{2}$$
$$= \sum_{g=1}^{G} \left\{ \frac{1}{|\mathcal{G}^{(g)}|} \sum_{t \in \mathcal{G}^{(g)}} \min_{\boldsymbol{s}^{(t)}} \left[-\mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) + \mu_1 \left|\left|\boldsymbol{s}^{(t)}\right|\right|_1 \right] + \mu_2 \left|\left|\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\right|\right|_{\mathsf{F}}^2 \right\} + \mu_3 \left|\left|\boldsymbol{L}\right|\right|_{\mathsf{F}}^2 \ ,$$

where $\boldsymbol{\theta}^{(t)} = \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\boldsymbol{L}\boldsymbol{s}^{(t)}$ and the L_1 norm of $\boldsymbol{s}^{(t)}$ is used to approximate the true vector sparsity. We employ regularization via the Frobenius norm $||\cdot||_{\mathsf{F}}$ to avoid overfitting on both the shared knowledge base $\boldsymbol{L}$ and each of the group projections $\boldsymbol{\Psi}^{(\mathcal{G}^{(1)})}, \ldots, \boldsymbol{\Psi}^{(\mathcal{G}^{(G)})}$. Note that this objective is closely related to PG-ELLA [Bou Ammar *et al.*, 2014], with the critical difference that it incorporates cross-domain transfer.

5.2 Online Solution to Cross-Domain MTL

Although Equation 2 allows for batch cross-domain MTL, the dependence on *all* available trajectories from all tasks (via $\mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) = \int_{\boldsymbol{\tau} \in \mathbb{T}^{(t)}} p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})\, \mathfrak{R}^{(t)}(\boldsymbol{\tau})\, d\boldsymbol{\tau}$) make the batch approach unsuitable for learning tasks consecutively, since the learner requires all trajectories for acquiring a successful behavior. Here, we derive an approximate optimization algorithm that is more suitable for lifelong learning.

Standardizing the Objective

To derive the approximate optimization problem, we note that policy gradient methods maximize the lower bound of $\mathcal{J}(\boldsymbol{\theta})$. In order to use Equation 2 for lifelong cross-domain transfer with policy gradients, we must first incorporate this lower bound into our objective function. Rewriting the error term in Equation 2 in terms of the lower bound yields

$$e_{\mathsf{T}}\left(\boldsymbol{L}, \boldsymbol{\Psi}^{(\mathcal{G}^{(1)})}, \ldots, \boldsymbol{\Psi}^{(\mathcal{G}^{(G)})}\right) \tag{3}$$
$$= \sum_{g=1}^{G} \left\{ \frac{1}{|\mathcal{G}^{(g)}|} \sum_{t \in \mathcal{G}^{(g)}} \min_{\boldsymbol{s}^{(t)}} \left[-\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right) + \mu_1 \left|\left|\boldsymbol{s}^{(t)}\right|\right|_1 \right] + \mu_2 \left|\left|\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\right|\right|_{\mathsf{F}}^2 \right\} + \mu_3 \left|\left|\boldsymbol{L}\right|\right|_{\mathsf{F}}^2 \ ,$$

with $\tilde{\boldsymbol{\theta}}^{(t)} = \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\boldsymbol{L}\boldsymbol{s}^{(t)}$. However, we can note that

$$\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right) \propto -\int_{\boldsymbol{\tau} \in \mathbb{T}^{(t)}} p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})\, \mathfrak{R}^{(t)}(\boldsymbol{\tau}) \log \left[\frac{p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})\, \mathfrak{R}^{(t)}(\boldsymbol{\tau})}{p_{\tilde{\boldsymbol{\theta}}^{(t)}}(\boldsymbol{\tau})} \right] d\boldsymbol{\tau} \ .$$

Therefore, maximizing the lower bound of $\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right)$ is equivalent to the following *minimization* problem:

$$\min_{\tilde{\boldsymbol{\theta}}^{(t)}} \int_{\boldsymbol{\tau} \in \mathbb{T}^{(t)}} p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})\, \mathfrak{R}^{(t)}(\boldsymbol{\tau}) \log \left[\frac{p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})\, \mathfrak{R}^{(t)}(\boldsymbol{\tau})}{p_{\tilde{\boldsymbol{\theta}}^{(t)}}(\boldsymbol{\tau})} \right] d\boldsymbol{\tau} \ , \tag{4}$$

which can be plugged into Equation 3 in place of $\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right)$ to obtain the MTL policy gradients objective.

Approximate Learning Objective

To eliminate the dependence of the objective function on all available trajectories, we can approximate $e_{\mathsf{T}}(\cdot)$ by performing a second-order Taylor expansion of $\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right)$ around the optimal solution $\boldsymbol{\alpha}^{(t)}$ to Equation 4, following the technique used in PG-ELLA [Bou Ammar *et al.*, 2014]. Note that attaining $\boldsymbol{\alpha}^{(t)}$ corresponds to solving the policy gradient problem of task $t \in \mathcal{G}^{(g)}$, which might be computationally expensive. Therefore, rather than using the above, we use an approximation acquired by performing a gradient step in task t: $\boldsymbol{\alpha}^{(t)} = \boldsymbol{\theta} + \eta \boldsymbol{\mathcal{I}}^{-1} \nabla_{\tilde{\boldsymbol{\theta}}^{(t)}} \mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right)$, where $\boldsymbol{\mathcal{I}}$ is the Fisher information matrix. The first derivative, needed for the second-order Taylor expansion, is given by:

$$\nabla_{\tilde{\boldsymbol{\theta}}^{(t)}} \mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right) = -\int_{\tau \in \mathbb{T}^{(t)}} p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})\, \mathfrak{R}^{(t)}(\boldsymbol{\tau})\, \nabla_{\tilde{\boldsymbol{\theta}}^{(t)}} \log p_{\tilde{\boldsymbol{\theta}}^{(t)}}(\boldsymbol{\tau})\, d\boldsymbol{\tau}$$

$$\log p_{\tilde{\boldsymbol{\theta}}^{(t)}}(\boldsymbol{\tau}) = \log p^{(t)}\left(\boldsymbol{x}_1^{(t)}\right) + \sum_{m=1}^{M^{(t)}} p^{(t)}\left(\boldsymbol{x}_{m+1}^{(t)} \mid \boldsymbol{x}_m^{(t)}, \boldsymbol{a}_m^{(t)}\right) + \sum_{m=1}^{M^{(t)}} \log \pi_{\tilde{\boldsymbol{\theta}}^{(t)}}\left(\boldsymbol{a}_m^{(t)} \mid \boldsymbol{x}_m^{(t)}\right) \ .$$

Therefore, we have that

$$\nabla_{\tilde{\boldsymbol{\theta}}^{(t)}} \mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right)$$
$$= -\int_{\boldsymbol{\tau} \in \mathbb{T}^{(t)}} p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})\, \mathfrak{R}^{(t)} \left[\sum_{m=1}^{M^{(t)}} \nabla_{\tilde{\boldsymbol{\theta}}^{(t)}} \log \pi_{\tilde{\boldsymbol{\theta}}^{(t)}}\left(\boldsymbol{a}_m^{(t)} | \boldsymbol{x}_m^{(t)}\right) \right] d\boldsymbol{\tau}$$
$$= -\mathbb{E}\left[\left(\sum_{m=1}^{M^{(t)}} \nabla_{\tilde{\boldsymbol{\theta}}^{(t)}} \log \pi_{\tilde{\boldsymbol{\theta}}^{(t)}}\left(\boldsymbol{a}_m^{(t)} | \boldsymbol{x}_m^{(t)}\right) \right) \mathfrak{R}^{(t)}(\boldsymbol{\tau}) \right] \ .$$

The second derivative of $\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)}\right)$ is then:

$$\boldsymbol{\Gamma}^{(t)} = -\mathbb{E}\left[\mathfrak{R}^{(t)}(\boldsymbol{\tau}) \sum_{m=1}^{M^{(t)}} \nabla^2_{\tilde{\boldsymbol{\theta}}^{(t)}, \tilde{\boldsymbol{\theta}}^{(t)}} \log \pi_{\tilde{\boldsymbol{\theta}}^{(t)}}\left(\boldsymbol{a}_m^{(t)} | \boldsymbol{x}_m^{(t)}\right) \right]_{\tilde{\boldsymbol{\theta}}^{(t)} = \boldsymbol{\alpha}^{(t)}} \ .$$

Substituting the second-order Taylor expansion yields the following approximation of Equation 2:

$$\hat{e}_{\mathsf{T}}\left(\boldsymbol{L}, \boldsymbol{\Psi}^{(\mathcal{G}^{(1)})}, \ldots, \boldsymbol{\Psi}^{(\mathcal{G}^{(G)})}\right) \tag{5}$$
$$= \sum_{g=1}^{G} \frac{1}{|\mathcal{G}^{(g)}|} \sum_{t \in \mathcal{G}^{(g)}} \min_{\boldsymbol{s}^{(t)}} \left[\left|\left|\boldsymbol{\alpha}^{(t)} - \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\boldsymbol{L}\boldsymbol{s}^{(t)}\right|\right|_{\boldsymbol{\Gamma}^{(t)}}^2 + \mu_1 \left|\left|\boldsymbol{s}^{(t)}\right|\right|_1 + \mu_2 \left|\left|\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}\right|\right|_{\mathsf{F}}^2 \right] + \mu_3 \left|\left|\boldsymbol{L}\right|\right|_{\mathsf{F}}^2 \ ,$$

where $||\boldsymbol{v}||_{\boldsymbol{A}}^2 = \boldsymbol{v}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{v}$, the constant term was suppressed as it has no effect on the minimization, and the linear term was ignored by construction since $\boldsymbol{\alpha}^{(t)}$ is a minimizer. Critically, we have eliminated the dependence on all available trajectories, making the problem suitable for an online MTL setting.

Learning the Policy

We fit the policy parameters in two steps. Upon observing a task $t \in \mathcal{G}^{(g)}$, we use gradient descent to update the shared repository $\boldsymbol{L}$ and the group projections $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}$. The update rules, acquired by taking the derivative of Equation 5 with respect to $\boldsymbol{L}$ and $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}$, can be written as:

$$\Delta \boldsymbol{L} = \eta_{\boldsymbol{L}} \Bigg[\sum_{g} \frac{1}{|\mathcal{G}^{(g)}|} \sum_{t \in \mathcal{G}^{(g)}} -\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})^{\mathsf{T}}} \boldsymbol{\Gamma}^{(t)} \boldsymbol{\alpha}^{(t)} \boldsymbol{s}^{(t)^{\mathsf{T}}} + \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})^{\mathsf{T}}} \boldsymbol{\Gamma}^{(t)} \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \boldsymbol{L} \boldsymbol{s}^{(t)} \boldsymbol{s}^{(t)^{\mathsf{T}}} + \mu_3 \boldsymbol{L} \Bigg] \tag{6}$$

$$\Delta \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} = \eta_{\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}} \Bigg[\frac{1}{|\mathcal{G}^{(g)}|} \sum_{t \in \mathcal{G}^{(g)}} -\boldsymbol{\Gamma}^{(t)} \boldsymbol{\alpha}^{(t)} \left(\boldsymbol{L} \boldsymbol{s}^{(t)} \right)^{\mathsf{T}} + \boldsymbol{\Gamma}^{(t)} \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \left(\boldsymbol{L} \boldsymbol{s}^{(t)} \right) \left(\boldsymbol{L} \boldsymbol{s}^{(t)} \right)^{\mathsf{T}} + \mu_2 \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \Bigg] , \tag{7}$$

where $\eta_{\boldsymbol{L}}$ and $\eta_{\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}}$ are the learning rates for the shared repository $\boldsymbol{L}$ and group projections $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})}$, respectively.

Having learned the shared representation $\boldsymbol{L}$ and group projections, task-specific coefficients $\boldsymbol{s}^{(t)}$ are then determined by solving an instance of Lasso [Tibshirani, 1996] to encode $\boldsymbol{\alpha}^{(t)}$ in the basis given by $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \boldsymbol{L}$, leading to a task-specific policy $\pi^{(t)} = p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{a} \mid \boldsymbol{x})$ where $\boldsymbol{\theta}^{(t)} = \boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \boldsymbol{L} \boldsymbol{s}^{(t)}$. For computationally demanding domains, we can update $\boldsymbol{s}^{(t)}$ less frequently, instead allowing the policy to change by altering $\boldsymbol{L}$ and the $\boldsymbol{\Psi}$'s. The complete implementation of our approach is available on the authors' websites.

6 Theoretical Guarantees

This section shows that our approach becomes stable as the number of tasks and groups grow large. Detailed proofs and definitions are provided in an online appendix available on the authors' websites. First, we consider the one-group setting by defining the following expected loss:[1]

$$\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|} \left(\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} | \mathcal{G}^{(g)} \right) = \mathbb{E}_{(\boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)})} \left[\min_{\boldsymbol{s}} l \left(\boldsymbol{L}, \boldsymbol{\Psi}^{\mathcal{G}^{(g)}}, \boldsymbol{s}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)} \middle| \mathcal{G}^{(g)} \right) \right] , \tag{8}$$

where the expectation is over each task t in group $\mathcal{G}^{(g)}$ according to the task's parameters $\left(\boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)}\right)$, and $l(\cdot)$ is the per-task loss of encoding $\boldsymbol{\alpha}^{(t)}$ in the basis given by $\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \boldsymbol{L}$.

Proposition 1.

$$\boldsymbol{\Psi}^{[|\mathcal{G}^{(g)}|, (\mathcal{G}^{(g)})]} - \boldsymbol{\Psi}^{[|\mathcal{G}^{(g)}|-1, (\mathcal{G}^{(g)})]} = \mathcal{O}\left(\frac{1}{|\mathcal{G}^{(g)}|} \right)$$

Proof. Here, we sketch the proof of Prop. 1. With $l \left(\boldsymbol{L}, \boldsymbol{\Psi}^{\mathcal{G}^{(g)}}, \boldsymbol{s}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)} \middle| \mathcal{G}^{(g)} \right)$, we can show that $\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|} \left(\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} | \mathcal{G}^{(g)} \right) - \hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|-1} \left(\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} | \mathcal{G}^{(g)} \right)$ is Lipschitz with $\mathcal{O}\left(\frac{1}{|\mathcal{G}^{(g)}|} \right)$. Further, given enough gradient steps, for $\boldsymbol{\Psi}^{(|\mathcal{G}^{(g)}|-1), (\mathcal{G}^{(g)})}$ to minimize $\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|-1} \left(\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} | \mathcal{G}^{(g)} \right)$, it is clear that change in the loss can be upper bounded by $2\lambda \left|\left| \boldsymbol{\Psi}^{(|\mathcal{G}^{(g)}|), (\mathcal{G}^{(g)})} - \boldsymbol{\Psi}^{(|\mathcal{G}^{(g)}|-1), (\mathcal{G}^{(g)})} \right|\right|_{\mathsf{F}}^2$. Combining the Lipschitz bound with previous facts concludes the proof. □

Proposition 2. *With $\boldsymbol{h}(\cdot)$ as the actual loss in $\mathcal{G}^{(g)}$, we show:*
1. $\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|} \left(\boldsymbol{\Psi}^{\mathcal{G}^{(g)}} | \mathcal{G}^{(g)} \right)$ converges a.s.
2. $\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|} \left(\boldsymbol{\Psi}^{\mathcal{G}^{(g)}} | \mathcal{G}^{(g)} \right) - \boldsymbol{h}_{|\mathcal{G}^{(g)}|} \left(\boldsymbol{\Psi}^{\mathcal{G}^{(g)}} | \mathcal{G}^{(g)} \right)$ converges a.s. to 0
3. $\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|} \left(\boldsymbol{\Psi}^{\mathcal{G}^{(g)}} | \mathcal{G}^{(g)} \right) - \boldsymbol{h} \left(\boldsymbol{\Psi}^{\mathcal{G}^{(g)}} | \mathcal{G}^{(g)} \right)$ converges a.s. to 0
4. $\boldsymbol{h} \left(\boldsymbol{\Psi}^{\mathcal{G}^{(g)}} | \mathcal{G}^{(g)} \right)$ converges a.s.

Proof. First, we show that the sum of positive variations of the stochastic process $\boldsymbol{u}_{|\mathcal{G}^{(g)}|} = \hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|} \left(\boldsymbol{\Psi}^{(|\mathcal{G}^{(g)}|), (\mathcal{G}^{(g)})} | \mathcal{G}^{(g)} \right)$ are bounded by invoking a corollary of the Donsker theorem [Van der Vaart, 2000]. This result in combination with a theorem from Fisk [1965], allows us to show that $u_{t\left(\mathcal{G}^{(g)}\right)}$ is a quasi-martingale that converges almost surely (a.s.). This fact along with a simple theorem of positive sequences allows us prove part 2 of the proposition. The final two parts (3 & 4) can be shown due to the equivalence of $\boldsymbol{h}$ and $\boldsymbol{h}_{|\mathcal{G}^{(g)}|}$ as $|\mathcal{G}^{(g)}| \to \infty$. □

Proposition 3. *The distance between $\boldsymbol{\Psi}^{(|\mathcal{G}^{(g)}|), (\mathcal{G}^{(g)})}$ and the set of $\boldsymbol{h}$'s stationary points converges a.s. to 0 as $|\mathcal{G}^{(g)}| \to \infty$.*

Proof. Both the surrogate $\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|}$ and the expected cost $\boldsymbol{h}$ have gradients that are Lipschitz with constant independent of $|\mathcal{G}^{(g)}|$. This fact, in combination with the fact that $\hat{\boldsymbol{h}}_{|\mathcal{G}^{(g)}|}$ and $\boldsymbol{g}$ converges a.s. as $|\mathcal{G}^{(g)}| \to \infty$, completes the proof. □

Next, we consider the loss of multiple groups:

$$\hat{\boldsymbol{g}}^{(\mathcal{G}^{(g)})} (\boldsymbol{L}) = \mathbb{E}_{\mathcal{G}^{(g)}} \left[\hat{\boldsymbol{h}}^{(|\mathcal{G}^{(g)}|)} \left(\boldsymbol{\Psi}^{\mathcal{G}^{(g)}} \right) \middle| \mathcal{G}^{(g)} \right] . \tag{9}$$

Proposition 4.

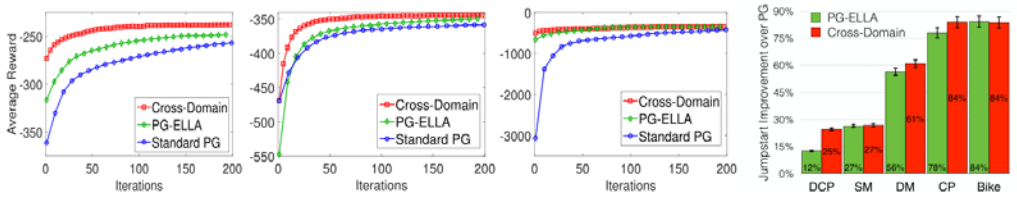
$$\boldsymbol{L}^{(\mathcal{G}^{(g)})} - \boldsymbol{L}^{(\mathcal{G}^{(g)})-1} = \mathcal{O}\left(\sum_{g=1}^{G} \frac{1}{|\mathcal{G}^{(g)}|} \right)$$

Proof. This can be easily shown as the upper bound of $\sum_g \left(\hat{\boldsymbol{h}} \left(\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \middle| \mathcal{G}^{(g)} \right) \cdot \right)$ is the sum of the bounds over $\hat{\boldsymbol{h}} \left(\boldsymbol{\Psi}^{(\mathcal{G}^{(g)})} \middle| \mathcal{G}^{(g)} \right)$. □

Proposition 5. *With $\boldsymbol{g}(\cdot)$ as the actual loss, we show:*
1. $\hat{\boldsymbol{g}}^{(\mathcal{G}^{(g)})} (\boldsymbol{L})$ converges a.s.
2. $\hat{\boldsymbol{g}}^{(\mathcal{G}^{(g)})} (\boldsymbol{L}) - \boldsymbol{g}^{(\mathcal{G}^{(g)})} (\boldsymbol{L})$ convergence a.s. to 0
3. $\hat{\boldsymbol{g}}^{(\mathcal{G}^{(g)})} (\boldsymbol{L}) - \boldsymbol{g}^{(\mathcal{G}^{(g)})} (\boldsymbol{L})$ convergence a.s. to 0
4. $\boldsymbol{g}^{(\mathcal{G}^{(g)})} (\boldsymbol{L})$ converges a.s.

Proof. This can be attained similarly to that of Prop. 2. □

[1] We super- or subscript variables with $(|\mathcal{G}^{(g)}|)$ to denote the version of the variable learned from $|\mathcal{G}^{(g)}|$ tasks in $\mathcal{G}^{(g)}$.



(a) Double Cart-Pole (b) Helicopter (c) Cart-Pole (d) Jumpstart over PG

Figure 2: Learning performance after interleaved training over multiple task domains. Figures (a) and (b) depict task domains where cross-domain transfer has a significant impact, showing that our approach outperforms standard PG and PG-ELLA. Figure (c) demonstrates that even when a domain benefits less from cross-domain transfer, our approach still achieves equivalent performance to PG-ELLA. Figure (d) depicts the average improvement in initial task performance over PG from transfer.

7 Empirical Results

We evaluated the ability of our approach to learn optimal control policies for multiple consecutive tasks from six different dynamical systems (each corresponding to one task domain):

Simple Mass (SM): The spring-mass-damper system is characterized by three parameters: spring and damping constants and the mass. The system's state is given by the position and velocity of the mass, which varies according to linear force. The goal is to control the mass to be in a specific state.

Double Mass (DM): The double spring-mass-damper has six parameters: two spring constants, damping constants, and masses. The state is given by the position and velocity of both masses. The goal is to control the first mass to a specific state, while only applying a linear force to the second.

Cart-Pole (CP): The dynamics of the inverted pendulum system are characterized by the cart's and pole's masses, the pole's length, and a damping parameter. The state is characterized by the cart's position and velocity, and the pole's angle and angular velocity. The goal is to balance the pole upright.

Double Cart-Pole (DCP): The DCP adds a second inverted pendulum to the CP system, with six parameters and six state features. The goal is to balance both poles upright.

Bicycle (Bike): The Bike model assumes a fixed rider, and is characterized by eight parameters. The goal is to keep the bike balanced as it rolls along the horizontal plane.

Helicopter (HC): This linearized model of a CH-47 tandem-rotor helicopter assumes horizontal motion at 40 knots. The main goal is to stabilize the helicopter by controlling the collective and differential rotor thrust.

For each of these systems, we created three different tasks by varying the system parameters to create systems with different dynamics, yielding 18 tasks total. These tasks used a reward function typical for optimal control, given by $-\sqrt{(x_m - \dot{x})^\intercal (x_m - \dot{x})} - \sqrt{a_m^\intercal a_m}$ where $\dot{x}$ is the goal state. Each round of the lifelong learning experiment, one task t was chosen randomly with replacement, and task t's model was trained from 100 sampled trajectories of length 50. This process continued until all tasks were seen at least once.

We then compared the performance of cross-domain lifelong learning with PG-ELLA and standard policy gradients (PG), averaging results over ∼94 trials per domain (each of which contained ∼60 interleaved training rounds). As the base PG learner in all algorithms, we used Natural Actor Critic [Peters & Schaal, 2008]. All regularization parameters (the μ's) were set to e^{-5}, and the learning rates and latent dimensions were set via cross-validation over a few tasks.

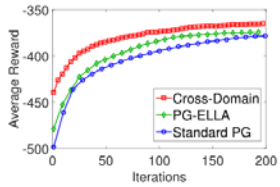


Figure 3: Average learning performance on a novel task domain (helicopter) after lifelong learning on other domains.

Figure 2 shows the average learning performance on individual domains after this process of interleaved lifelong learning, depicting domains in which cross-domain transfer shows clear advantages over PG-ELLA and PG (e.g., DCP, HC), and an example domain where cross-domain transfer is less effective (CP). Note that even in a domain where cross-domain transfer provides little benefit, our approach achieves equivalent performance to PG-ELLA, showing that cross-domain transfer does not interfere with learning effectiveness. On all task domains except HC, our approach provides a significant increase in initial performance due to transfer (Figure 2(d)).

Cross-domain transfer provides significant advantages when the lifelong learning agent faces a novel task domain. To evaluate this, we chose the most complex of the task domains (helicopter) and trained the lifelong learner on tasks from all other task domains to yield an effective shared knowledge base $\boldsymbol{L}$. Then, we evaluated the agent's ability to learn a new task from the helicopter domain, comparing the benefits of cross-domain transfer from $\boldsymbol{L}$ with PG-ELLA and PG (both of which learn from scratch on the new domain). Figure 3 depicts the result of learning on a novel domain, averaged over ten trials for all three HC tasks, showing the effectiveness of cross-domain lifelong learning in this scenario.

8 Conclusion

We have presented the first lifelong RL method that supports autonomous and efficient cross-domain transfer. This approach provides a variety of theoretical guarantees, and can learn effectively across multiple task domains, providing improved performance over single-domain methods.

Acknowledgements

This research was supported by ONR grant #N00014-11-1-0139 and AFRL grant #FA8750-14-1-0069. We thank the anonymous reviewers for their helpful feedback.

References

[Bou Ammar *et al.*, 2014] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, & Matthew Taylor. Online multi-task learning for policy gradient methods. In *International Conference on Machine Learning (ICML)*, 2014.

[Bou Ammar *et al.*, 2012] Haitham Bou Ammar, Mathew E. Taylor, Karl Tuyls, Kurt Driessens, & Gerhard Weiss. Reinforcement learning transfer via sparse coding. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.

[Bou Ammar *et al.*, 2013] Haitham Bou Ammar, Decebal Constantin Mocanu, Matthew Taylor, Kurt Driessens, Gerhard Weiss, & Karl Tuyls. Automatically mapped transfer between reinforcement learning tasks via three-way restricted Boltzmann machines. In *European Conference on Machine Learning (ECML)*, 2013.

[Bertsekas, 1995] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, vol. 1. Athena Scientific, Belmont, MA, 1995.

[Busoniu *et al.*, 2008] Lucian Busoniu, Robert Babuska, & Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, 2008.

[Deisenroth *et al.*, 2014] Marc Peter Deisenroth, Peter Englert, Jan Peters, & Dieter Fox. Multi-task policy search for robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3876–3881, 2014.

[Dempster & Leemans, 2006] Michael A.H. Dempster & Vasco Leemans. An automated FX trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552, 2006.

[Fisk, 1965] D.L. Fisk. Quasi-martingales. *Transactions of the American Mathematical Society*, 120(3):369–389, 1965.

[Han *et al.*, 2012] Shaobo Han, Xuejun Liao, & Lawrence Carin Cross-domain multi-task learning with latent probit models. In *International Conference on Machine Learning (ICML)*, 2012.

[Kang *et al.*, 2011] Zhuoliang Kang, Kristen Grauman, & Fei Sha. Learning with whom to share in multi-task feature learning. In *International Conference on Machine Learning (ICML)*, pages 521–528, 2011.

[Kober & Peters, 2011] Jens Kober & Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2), 2011.

[Konidaris & Doshi-Velez, 2014] George Konidaris & Finale Doshi-Velez. Hidden parameter Markov decision processes: an emerging paradigm for modeling families of related tasks. In the *AAAI Fall Symposium on Knowledge, Skill, and Behavior Transfer in Autonomous Robots*, 2014.

[Kumar & Daumé III, 2012] Abhishek Kumar & Hal Daumé III. Learning task grouping and overlap in multi-task learning. In *International Conference on Machine Learning (ICML)*, 2012.

[Lazaric & Ghavamzadeh, 2010] Alessandro Lazaric & Mohammad Ghavamzadeh. Bayesian multi-task reinforcement learning. In *International Conference on Machine Learning*, pages 599–606, 2010.

[Li *et al.*, 2009] Hui Li, Xuejun Liao, & Lawrence Carin. Multi-task reinforcement learning in partially observable stochastic environments. *Journal of Machine Learning Research*, 10:1131–1186, 2009.

[Maurer *et al.*, 2013] Andreas Maurer, Massimiliano Pontil, & Bernardino Romera-Paredes. Sparse coding for multitask and transfer learning. In *International Conference on Machine Learning*, pages 343–351, 2013.

[Peters & Schaal, 2008] Jan Peters & Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008.

[Peters *et al.*, 2005] Jan Peters, Sethu Vijayakumar, & Stefan Schaal. Natural actor-critic. In *European Conference on Machine Learning (ECML)*, pages 280–291, 2005.

[Rückstieß *et al.*, 2008] Thomas Rückstieß, Martin Felder, & Jürgen Schmidhuber. State-dependent exploration for policy gradient methods. In *Machine Learning and Knowledge Discovery in Databases*, pages 234–249, 2008.

[Ruvolo & Eaton, 2013] Paul Ruvolo & Eric Eaton. ELLA: an efficient lifelong learning algorithm. In *International Conference on Machine Learning (ICML)*, 2013.

[Smart & Kaelbling, 2002] William D. Smart & Leslie Pack Kaelbling. Effective reinforcement learning for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3404–3410, 2002.

[Snel & Shimon Whiteson, 2014] Matthijs Snel & Shimon Whiteson. Learning potential functions and their representations for multi-task reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 28(4):637–681, 2014.

[Sutton *et al.*, 1999] Richard S. Sutton, David A. McAllester, Satinder P. Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems* 99:1057–1063, 1999.

[Taylor & Stone, 2009] Matthew E. Taylor & Peter Stone. Transfer learning for reinforcement learning domains: a survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[Taylor *et al.*, 2007] Matthew E. Taylor, Shimon Whiteson, & Peter Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 156–163, 2007.

[Taylor *et al.*, 2008] Matthew E. Taylor, Gregory Kuhlmann, & Peter Stone. Autonomous transfer for reinforcement learning. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 283–290, 2008.

[Thrun & O'Sullivan, 1996] Sebastian Thrun & Joseph O'Sullivan. Discovering structure in multiple learning tasks: the TC algorithm. In *International Conference on Machine Learning*, 1996.

[Tibshirani, 1996] Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[Van der Vaart, 2000] A.W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge University Press, 2000.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[Wilson *et al.*, 2007] Aaron Wilson, Alan Fern, Soumya Ray, & Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *International Conference on Machine Learning (ICML)*, 2007.

Appendix E: Zhan et al., 2016

Theoretically-Grounded Policy Advice from Multiple Teachers in Reinforcement Learning Settings with Applications to Negative Transfer

Yusen Zhan[1], **Haitham Bou Ammar**[2], **and Matthew E. Taylor**[1]
[1]Washington State University, Pullman, Washington
[2]Princeton University, Princeton, New Jersey
yusen.zhan@wsu.edu, hammar@princeton.edu, taylorm@eecs.wsu.edu

Abstract

Policy advice is a transfer learning method where a student agent is able to learn faster via advice from a teacher. However, both this and other reinforcement learning transfer methods have little theoretical analysis. This paper formally defines a setting where multiple teacher agents can provide advice to a student and introduces an algorithm to leverage both autonomous exploration and teacher's advice. Our regret bounds justify the intuition that good teachers help while bad teachers hurt. Using our formalization, we are also able to quantify, for the first time, when negative transfer can occur within such a reinforcement learning setting.

1 Introduction

Reinforcement Learning (RL) has become a popular framework for autonomous behavior generation from limited feedback [Sutton and Barto, 1998]. Typical RL methods learn in isolation increasing their learning times and sample complexities. Transfer learning aims to significantly improve learning by providing informative knowledge from an external source. The source of such knowledge varies from source agents to humans providing advice [Erez and Smart, 2008; Taylor and Stone, 2009]. In this paper, we focus on a framework referred to as action advice or the *advice model* [Taylor *et al.*, 2014]. Here, the agent (i.e., student), learning in a task, has access to a teacher (another agent or human) which can provide action suggestions to facilitate learning. Given "good-enough" teachers, such advice models have shown multiple benefits over standard RL techniques. For example, others [Taylor *et al.*, 2014; Zimmer *et al.*, 2014] show reduced learning times and sample complexities for successful behavior.

These methods, however, suffer from two main drawbacks. First, validation results are empirical in nature and not formally-grounded. We do not have fundamental understanding of these methods. Consequently, it is difficult to formally comprehend why these methods work. Second, most of these techniques require the availability of a "good-enough" (optimal) teacher to benefit the student. Unfortunately, access to such teachers is difficult in a variety of complex domains, reducing the applicability of policy advice in real-world settings.

In this paper, we remedy the aforementioned drawbacks by proposing a new framework for policy advice. Our method formally generalizes current single-teacher advice models to the multi-teacher setting. Our algorithm also remedies the need for optimal teachers by exploiting both the student's and the teacher's knowledge. Even if the teacher is not optimal, a student, using our algorithm, is still capable of acquiring optimal behavior in a task; a property not supported by some state-of-the-art methods, e.g., learning from demonstration. We theoretically and empirically analyze the performance of the proposed method and derive, for the first time, regret bounds quantifying the successfulness of action advice. We also provide theoretical justification for current methods (i.e., single-teacher models) as special case of our formulation[1]. Our contributions can be summarized as:

- defining (formally) multi-teacher advice models,
- introducing novel algorithms leveraging teacher and student knowledge,
- deriving the regret analysis showing reduced sample complexities,
- deriving theoretical guarantees for single teacher advice models, and
- quantifying negative transfer under such advice model.

Interestingly, these theoretical results justify a well-known intuition inherent to advice models: "good teachers help while bad teachers hurt." The results show that students can still achieve optimal behavior when being advised by bad teachers. They, however, pay an extra cost in terms of their learning times or sample complexities, relative to an optimal teacher. This should inspire researchers to adopt high quality teacher policies or avoid "bad teachers" if possible.

Given our formalization, we also derive a relation to negative transfer. We quantify, for the first time, the occurrence of negative transfer in action advice models, shedding the light on failure modes of these methods. Consequently, these results yield two claims about transfer learning. First, high quality transfer knowledge may still cause negative transfer

[1]The full version: https://arxiv.org/pdf/1604.03986v1

when the target algorithm is able to outperform the source knowledge. Second, expert knowledge is important for the researchers to determine whether or not to transfer because evaluation of the transfer knowledge is usually expensive (it is equivalent to evaluating the teacher policy in the target MDP).

2 Preliminaries

2.1 Online Reinforcement Learning & Regret Model

In RL, an agent must sequentially select actions to maximize its total expected return. Such problems are formalized as a Markov decision Process (MDP), defined as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the finite state and actions spaces with a total size of $|\mathcal{S}|$ and $|\mathcal{A}|$ respectively, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the probability transition kernel describing the task dynamics, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function quantifying the performance of the agent. The total expected return of an agent following an algorithm, $\mathcal{G}$, to compute the optimal action-selection rule from a starting state $\boldsymbol{s} \in \mathcal{S}$ after T time steps is defined as:

$$\mathcal{R}^{\mathcal{G}}(\boldsymbol{s}, T) = \mathbb{E}\left[\sum_{t=0}^{T} \mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right], \tag{1}$$

with $\boldsymbol{s}_t \in \mathcal{S}$ and $\boldsymbol{a}_t \in \mathcal{A}$. The goal is to determine an optimal policy, $\pi^\star : \mathcal{S} \to \mathcal{A}$ that maximizes the total expected return.

Regret Model: Similar to standard online learning, we quantify the performance of the algorithm, $\mathcal{G}$, by measuring its regret with respect to optimality. We define the regret of a state $\boldsymbol{s}$ after T time steps in terms of the expected reward as:

$$\boldsymbol{\Delta}^{\mathcal{G}}(\boldsymbol{s}, T) = \boldsymbol{\lambda}^\star T - \mathcal{R}^{\mathcal{G}}(\boldsymbol{s}, T), \tag{2}$$

where $\boldsymbol{\lambda}^\star$ is the optimal reward acquired by following an optimal algorithm $\mathcal{G}^\star$ at each time step. In the general case when no reachability assumptions are imposed, it is easy to construct MDPs in which algorithms suffer high regret. Following Puterman [2005], we remedy this problem by considering weakly-communicating MDPs[2] defined as follows.

Definition 1. *An MDP is called weakly communicating in such a case where the state set $\mathcal{S}$ can be decomposed into two subsets, $\mathcal{S}_1$ and $\mathcal{S}_2$. In $\mathcal{S}_1$ any state is reachable from every other state under a deterministic policy, π, while states in $\mathcal{S}_2$ are transient under all policies.*

The optimal gain, $\boldsymbol{\lambda}^\star$ in Equation 2, is state independent. That is, any $\boldsymbol{s} \in \mathcal{S}$, shares the same optimal expected reward [Puterman, 2005], which can be solved for using:

$$\boldsymbol{h}^\star + \boldsymbol{\lambda}^\star \boldsymbol{e} = \max_{\boldsymbol{a} \in \mathcal{A}} \{\mathcal{R}(\boldsymbol{s}, \boldsymbol{a}) + \mathcal{P}_{\boldsymbol{s},\boldsymbol{a}}^{\mathsf{T}} \boldsymbol{h}^\star\},$$

where $\boldsymbol{h}^\star$ is an $|\mathcal{S}|$ dimensional vector typically referred to as the bias vector, $\mathcal{P}_{\boldsymbol{s},\boldsymbol{a}}$ denotes the probability to transition from $\boldsymbol{s}$ applying action $\boldsymbol{a}$, and $\boldsymbol{e} \in \mathbb{R}^{|\mathcal{S}|}$ is a unit dimensional vector. When needed, we explicitly write the dependency of $\boldsymbol{\lambda}^\star$ and $\boldsymbol{h}^\star$ as $\boldsymbol{h}^\star(\boldsymbol{s}; \mathcal{M})$ and $\boldsymbol{\lambda}^\star(\mathcal{M})$. We also define the span of $\boldsymbol{h}$ as: $\text{sp}(\boldsymbol{h}) = \max_{\boldsymbol{s} \in \mathcal{S}} \boldsymbol{h}(\boldsymbol{s}) - \min_{\boldsymbol{s} \in \mathcal{S}} \boldsymbol{h}(\boldsymbol{s})$.

[2]Please note that weakly-communicating MDPs are considered the most general among subclasses of MDPs, see Puterman [2005].

Algorithm 1 REGAL.C: Constrained Optimization

Input: parameter H, dataset $\mathcal{D}_i$ and current time T
Output: $\hat{\pi}_{i+1}$

1: $t_i =$ current time T
2: Use $\mathcal{D}_i$ to update the state transition probabilities by

$$\hat{P}^t_{\boldsymbol{s},\boldsymbol{a}}(\boldsymbol{s}') = \frac{N(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; t)}{\max\{N(\boldsymbol{s}, \boldsymbol{a}; t), 1\}} \tag{3}$$

3: With $t = t_i$, $\mathcal{M}^i$ is the set of MDPs s.t.

$$\left\| P_{\boldsymbol{s},\boldsymbol{a}} - \hat{P}^t_{\boldsymbol{s},\boldsymbol{a}} \right\|_1 \leq \sqrt{\frac{12|\mathcal{S}| \log(2|\mathcal{A}|t/\delta)}{\max\{N(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}; t), 1\}}}$$

4: Select $M^i \in \mathcal{M}^i$ by following optimization equation over $\forall M \in \mathcal{M}^i$,

$$\max \boldsymbol{\lambda}^\star(M) \text{ s.t. } \text{sp}(\boldsymbol{h}^*(\mathcal{M})) \leq H$$

5: $\hat{\pi}_{i+1}$=average reward optimal policy for M^i (value iteration)
6: **return** $\hat{\pi}_{i+1}$

Finally, we follow Bartlett and Tewari [2009] to define reachability in weakly communicating MDPs using the *one-way diameter*: $\text{diam}_{\text{one-way}}(\mathcal{M}) = \max_{\boldsymbol{s} \in \mathcal{S}} \min_\pi T^\pi_{\boldsymbol{s}_1 \to \bar{\boldsymbol{s}}}$, with $T^\pi_{\boldsymbol{s}_1 \to \bar{\boldsymbol{s}}}$ being the expected number of steps needed for reaching $\bar{\boldsymbol{s}} = \arg\max_{\boldsymbol{s} \in \mathcal{S}} \boldsymbol{h}^\star(\boldsymbol{s}; \mathcal{M})$ from $\boldsymbol{s}_1 \in \mathcal{S}$.

2.2 Algorithms for Weakly-Communicating MDPs

REGAL.C is an on-line algorithm for weakly communicating MDPs developed by Bartlett and Tewari [2009]. The basic idea is that the REGAL.C can estimate the true MDP with high probability in order to learn an ϵ-optimal policy with high probability. Let $N(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; t)$ be the number of state-action-state triples $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ that have been visited at time t. Further, let t_i to denote the initial time of the iteration i. For brevity, we use $N_i(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ and $N_i(\boldsymbol{s}, \boldsymbol{a})$ to denote $N(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; t_i)$ and $N(\boldsymbol{s}, \boldsymbol{a}; t_i)$ at iteration i. We also use $v_i(\boldsymbol{s}, \boldsymbol{a}) = N_{i+1}(\boldsymbol{s}, \boldsymbol{a}) - N_i(\boldsymbol{s}, \boldsymbol{a})$ to denote the number of times a state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ is visited during iteration i. For each iteration i, REGAL acquires a dataset $\mathcal{D}_i$ as input and updates the transition probability (see Equation 3). It then constructs a set of MDPs $\mathcal{M}^i$ to select from using $\max \lambda^*(M)$ s.t. $sp(h^*(M)) \leq H$, where H is the upper bound on the span $\text{sp}(\boldsymbol{h}^*(\mathcal{M}))$. Given the MDP, REGAL.C uses value iteration for acquiring the optimal policy. These steps are summarized in Algorithm 1.

2.3 Single Teacher Advice Model

The single teacher advice model is a framework in which a student learning in an environment benefits from a teacher's advice to speed-up learning. We define such a framework as the tuple of $\langle \pi^{\mathfrak{T}}, \boldsymbol{b}, \mathfrak{S}, \boldsymbol{f}_d \rangle$. Here, $\pi^{\mathfrak{T}}$ denotes the teacher's policy, $\boldsymbol{b}$ represents the budget constraining the teacher's advice, $\mathfrak{S}$ is the student, and $\boldsymbol{f}_d$ is a function controlling the advice from the teacher to the student. Apart from considering single teacher models, previous work assumed optimal teachers where students always execute recommended actions. It is

easy to construct complex settings in which access to optimal teachers is difficult. Consequently, we extend these works to the more realistic settings of sub-optimal teachers, as we detail later.

3 Multiple Teacher Advice Model

In this section we start by extending the single teacher model of Taylor *et al.* [2014] to the multiple *non-optimal* teacher setting. Our advice model for m teachers is defined as the tuple $\langle \Pi, \mathcal{B}, \mathfrak{S}, \boldsymbol{f}_d \rangle$, where $\Pi = \{\pi_1^{\mathfrak{T}_1}, \pi_2^{\mathfrak{T}_2}, \ldots, \pi_m^{\mathfrak{T}_m}\}$ is the set of $m \in \mathbb{N}$ teacher policies, and $\mathcal{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m\}$ denotes the set of budgets. It is easy to see that in case $\Pi = \{\pi^{\mathfrak{T}}\}$ and $\mathcal{B} = \{\boldsymbol{b}\}$, we can easily recover the special case single teacher model. We also generalize the work of Taylor *et al.* [2014] by making no restrictive assumptions on the optimality of any of the teachers. We measure the performance of the teacher with respect to a base policy $\pi^{\mathfrak{B}}$ in terms of regret:

Definition 2. *Given a teacher's policy, $\pi^{\mathfrak{T}} \in \Pi$, and a base policy $\pi^{\mathfrak{B}}$, then the regret of following $\pi^{\mathfrak{T}}$ is related to that acquired by following $\pi^{\mathcal{B}}$ using:*

$$\boldsymbol{\Delta}^{\mathfrak{T}}(\boldsymbol{s}, T) = \rho \boldsymbol{\Delta}^{\mathfrak{B}}(\boldsymbol{s}, T),$$

where $\rho \geq 0$ denotes the regret ratio, $\boldsymbol{\Delta}^{\mathfrak{T}}(\boldsymbol{s}, T) = \boldsymbol{\lambda}^{\star} T - \mathcal{R}^{\mathfrak{T}}(\boldsymbol{s}, T)$ and $\boldsymbol{\Delta}^{\mathfrak{B}}(\boldsymbol{s}, T) = \boldsymbol{\lambda}^{\star} T - \mathcal{R}^{\mathfrak{B}}(\boldsymbol{s}, T)$.

The above definition captures the three interesting cases quantifying the performance of an advice-based algorithm. If the teacher is optimal, i.e., when $\boldsymbol{\Delta}^{\mathfrak{T}}(\boldsymbol{s}, T) = 0$, ρ is also 0. In case $0 < \rho \leq 1$, then $\boldsymbol{\Delta}^{\mathfrak{T}}(\boldsymbol{s}, T) \leq \boldsymbol{\Delta}^{\mathfrak{B}}(\boldsymbol{s}, T)$ indicating the the teacher's policy is at least as good as the base policy $\pi^{\mathfrak{B}}$. Finally, when $\rho > 1$, $\boldsymbol{\Delta}^{\mathfrak{T}}(\boldsymbol{s}, T) > \boldsymbol{\Delta}^{\mathfrak{B}}(\boldsymbol{s}, T)$ implying the underperformance of the teacher. Consequently, with the correct choice of the teacher by ρ one can still achieve successful advice even in such a generalized setting.

4 Efficient Multi-teacher Advice

In this section, we propose a new algorithm which combines the advice policy and the MDPs information collected so far. This allows for an accurate framework outperforming state-of-the-art techniques for policy advice. On a high level, our algorithm consists of three main steps. First, a combined policy is constructed based on multiple teachers. Second, data depending on both teacher's advice as well as MDP information is collected. Third, a new policy is computed online.

Next, we outline each of the three steps and describe our novel algorithm. Having achieved an accurate advice model, we then rigorously analyze the theoretical aspects of our method and show a decrease in sample complexities compared to current techniques.

4.1 The Grand-Teacher

Our method of policy advice constructs a grand teacher combining all teacher policies in a meta-policy. To construct the grand-teacher, we use an ensemble method and design two meta-policy variations: online and offline-constructions. Next, we detail each of the two variations.

Online Grand-Teacher: In the *online* construction, whenever the student observes an unvisited state, $\boldsymbol{s} \in \mathcal{S}$, each teacher provides its policy advice of the form $\pi_i^{\mathfrak{T}_i}$, for all $i \in \{1, \ldots, m\}$ with m being the total number of teachers. The student then selects and stores the majority action from all teachers for that state $\boldsymbol{s}$. As far as budget is concerned, it is easy to see that we only require to know advice for each state in $\mathcal{S}$, thus $\boldsymbol{b}_1 = \cdots = \boldsymbol{b}_m = |\mathcal{S}|$. Though easy to implement and test, the online construction suffers from the potentially unrealistic need for the continuous availability of online teachers.

Algorithm 2 Offline Construction of the Meta-Teacher

Input: The set of states in the MDP, $\mathcal{S}$.

1: **while** $\exists s \in \mathcal{S}$ is not visited **do**
2: Follow a policy in the MDP
3: **if** Current state $\boldsymbol{s}$ is not visited **then**
4: Query all teachers for advice and select action $\boldsymbol{a}$ using Majority Vote.
5: **return** $\pi^{\text{grand-teacher}}$

Offline Grand-Teacher: To eliminates the need for an online teacher at each visit of a new state, the offline procedure traverses the states in the MDP for constructing the meta-advice policy. The main steps of this construction is summarized in Algorithm 2.

Note that Algorithm 2 is capable of constructing an offline meta-teacher but requires extra exploration in the MDP. We next show that $\mathcal{O}\left(|\mathcal{S}| \log \frac{|\mathcal{S}|}{\boldsymbol{\delta}}\right)$ steps are enough to explore each state in the MDP with high probability:

Theorem 1 (Sample Complexity)**.** *If Algorithm 2 independently and uniformly explores each state $\boldsymbol{s} \in \mathcal{S}$, then with probability of at least $1 - \boldsymbol{\delta}$, $\mathcal{O}\left(|\mathcal{S}| \log \frac{|\mathcal{S}|}{\boldsymbol{\delta}}\right)$ steps are sufficient to visit each state at least one time.*

4.2 Multi-Teacher Advice Algorithm

To improve current methods and arrive at a more realistic advice framework, we now introduce our algorithm combining the grand-teacher's policy and information attained by the student from the MDP.

Our algorithm is based on the following intuition. At the beginning of the learning process, a student requires guidance as it typically has little to no information of the task to be solved. As time progress and the student explores, the MDP can be effectively exploited for successful learning. Unfortunately, such a process is not well modeled using current methods. Here, we remedy this problem by introducing an algorithm which follows the teacher's advice at the very beginning and then switches to a policy computed by an algorithm operating within the MDP. That is, the teacher guides the student at the beginning of the learning process and as the student gathers more experience, the teacher's influence diminishes over time by switching into a policy computed by REGAL.C. The overall procedure is summarized in Algorithm 3. Note that our algorithm is inspired by DAGGER [Ross *et al.*, 2010] in the sense that policies are updated by collecting data using a mixture of action selection rules (i.e., student and teacher policies). Contrary to DAGGER, however, our method col-

Algorithm 3

Input: $\pi^{\mathfrak{T}}$ = the grand-teacher policy, $\hat{\pi}_1$=any policy
Output: π, the ϵ-optimal policy

$T = 0$
for $i = 1$ to m **do**
  Let $\pi_{i+1} = \beta_i \pi^T + (1 - \beta_i)\hat{\pi}_i$
  Follow π_i until T_i-steps
  Get dataset D_i
  $T = T + T_i$
  $\hat{\pi}_{i+1} = REGAL.C(D_i, T)$ {See Algorithm 1}
return π_{m+1}

lects all trajectories opposed to only collecting inconsistent actions, allowing for more accurate and efficient updates.

To leverage both the teacher's and learned policies, we set a mixed policy of the form $\pi_{i+1} = \beta_i \pi^{\mathfrak{T}} + (1 - \beta_i)\hat{\pi}_i$, for $0 \leq \beta_i \leq 1$ to guide the student's dataset collection while allowing the teacher to fractionally control exploration needed to collect data at the next iteration. β should typically be set so as to decay exponentially over time. This decreases the student's reliance on the teacher and allows it to exploit the knowledge gathered from the MDP to learn better behaving policies than that of the teacher. It is for this reason that our algorithm, contrary to other methods, does not impose any optimality restrictions on the teacher. Having collected the dataset, Algorithm 3 uses REGAL.C (Algorithm 1) to update $\hat{\pi}_i$.

4.3 Theoretical Guarantees

In this section we formally derive the regret exhibited by our algorithm. At a high level, we provide two theoretical results. In the first, we consider the general teacher case, while in the second we derive a corollary of the regret for optimal teachers. We show, for the first time, *better than constant* improvements compared to standard learners.

Theorem 2. *Assume Algorithm 3 is running for total T steps in a weakly communicating MPD $\mathcal{M}$ starting from an initial state $\boldsymbol{s} \in \mathcal{S}$. Let H be a parameter such that $H \geq sp(\boldsymbol{h}^\star(\mathcal{M}))$. Then, with a probability of at least $1 - \delta$, the total regret is given by: $\boldsymbol{\Delta}(\boldsymbol{s}, T) = \mathcal{O}\left((1 - \boldsymbol{\beta} + \boldsymbol{\rho}\boldsymbol{\beta})H|\mathcal{S}|\sqrt{|\mathcal{A}|T \log \frac{|\mathcal{A}|T}{\delta}}\right)$, where $\boldsymbol{\beta} \in [0, 1]$ such that $1 - \boldsymbol{\beta} = \max_{1 \leq i \leq m}\{1 - \beta_i\}$, and $\boldsymbol{\rho} \geq 0$ is the ratio between the teacher's regret $\boldsymbol{\Delta}^{\mathfrak{T}}$ and the regret exhibited by REGAL.C $\boldsymbol{\Delta}^{REGAL.C}$ such that $\Delta^{\mathfrak{T}} \leq \boldsymbol{\rho}\boldsymbol{\Delta}^{REGAL.C}$.*

Proof. Due to the space limits, we provide a proof sketch. The proof is based on the regret bound of REGAL.C. We introduce the regret ratio to reduce the grand-teacher's regret to the REGAL.C's regret. Then, we apply the Hoeffding's inequality to arrive at the statement of the theorem. □

Theorem 2 implies that the teacher improves learning as long as it is "good." Namely, if $0 < \boldsymbol{\rho} \leq 1$, $1 - \boldsymbol{\beta} + \boldsymbol{\rho}\boldsymbol{\beta} \leq 1$, $\boldsymbol{\beta} \in [0, 1]$ which implies the student can enjoy a fraction of REGAL.C's regret. However, if $\boldsymbol{\rho} > 1$, $1 - \boldsymbol{\beta} + \boldsymbol{\rho}\boldsymbol{\beta} > 1$, the student suffers more regret than the original REGAL.C algorithm. This justifies our intuition that good teachers assist learning while poor ones hamper learning. Moreover, if there exists prior knowledge that a teacher has poor performance, it would be better off for the student to neglect its advice as it will suffer extra regret.

If the teacher's $\boldsymbol{\rho} = 0$, we have the following Corollary:

Corollary 1. *If the teacher is optimal, then with at least a probability of $1 - \delta$ the total regret is given by: $\boldsymbol{\Delta}(\boldsymbol{s}, T) = \mathcal{O}\left((1 - \boldsymbol{\beta})H|\mathcal{S}|\sqrt{|\mathcal{A}|T \log \frac{|\mathcal{A}|T}{\delta}}\right)$.*

Remark 1. *Please note that the above theoretical results are more than a constant improvement to the regret. Notice that $\boldsymbol{\beta}$ depends on the number of iterations which can be bounded by $|\mathcal{S}|$ and $|\mathcal{A}|$ of the input MDP $\mathcal{M}$ [Auer* et al.*, 2009]. Further, $\boldsymbol{\rho}$ depends on the input teacher's policy which is also an input to Algorithm 3. Consequently, it can be shown that these regret improvements exceed simple constant bounds.*

5 Negative Transfer

To formalize the relation to negative transfer, we recognize that the regret ratio can be written as:

$$\rho = \frac{\boldsymbol{\Delta}^{\mathfrak{T}}(\boldsymbol{s}, T)}{\boldsymbol{\Delta}^{\mathfrak{B}}(\boldsymbol{s}, T)} = \frac{\boldsymbol{\lambda}^\star T - \mathcal{R}^{\mathfrak{T}}(\boldsymbol{s}, T)}{\boldsymbol{\lambda}^\star T - \mathcal{R}^{\mathfrak{B}}(\boldsymbol{s}, T)} \tag{4}$$

This suggests that we can estimate the ratio by calculating $\boldsymbol{\lambda}^\star$ and $\mathcal{R}^\pi(\boldsymbol{s}, T)$, given a policy π. So, we use

$$\rho(\pi_1, \pi_2, T) = \frac{\boldsymbol{\lambda}^\star T - \mathcal{R}^{\pi_1}(\boldsymbol{s}, T)}{\boldsymbol{\lambda}^\star T - \mathcal{R}^{\pi_2}(\boldsymbol{s}, T)}$$

to denote the regret ratio between policy π_1 and π_2 until step T. At this stage, we define:

- Negative transfer from policy π_1 to π_2 until T steps: $\rho(\pi_1, \pi_2, T) > 1$.
- Positive transfer from policy π_1 to π_2 until T steps: $\rho(\pi_1, \pi_2, T) \leq 1$

To formalize negative transfer, our goal at this stage is to relate $\rho(\cdot)$ to a metric between source and target tasks. For that sake, we define: $d_t^s(\pi_s) = \hat{\mathcal{R}}_s^{\pi_s}(\boldsymbol{s}, T) - \hat{\mathcal{R}}_t^{\pi_s}(\boldsymbol{s}, T)$, with $\hat{\mathcal{R}}_s^{\pi_s}(\boldsymbol{s}, T)$ and $\hat{\mathcal{R}}_t^{\pi_s}(\boldsymbol{s}, T)$ being the agent's estimates of the rewards in the source and the target after T steps. Consequently, an estimate $\hat{\rho}$ to ρ can be derived as:

$$\begin{aligned}
&\hat{\rho}(\pi_s, \pi_t, T) \\
&= \frac{\boldsymbol{\lambda}^\star T - \hat{\mathcal{R}}_t^{\pi_s}(\boldsymbol{s}, T)}{\boldsymbol{\lambda}^\star T - \hat{\mathcal{R}}_t^{\pi_t}(\boldsymbol{s}, T)} \\
&= \frac{\boldsymbol{\lambda}^\star T + \left(\hat{\mathcal{R}}_s^{\pi_s}(\boldsymbol{s}, T) - \hat{\mathcal{R}}_t^{\pi_s}(\boldsymbol{s}, T)\right) - \hat{\mathcal{R}}_s^{\pi_s}(\boldsymbol{s}, T)}{\boldsymbol{\lambda}^\star T - \hat{\mathcal{R}}_t^{\pi_t}(\boldsymbol{s}, T)} \\
&= \frac{\boldsymbol{\lambda}^\star T + d_t^s(\pi_s) - \hat{\mathcal{R}}_s^{\pi_s}(\boldsymbol{s}, T)}{\boldsymbol{\lambda}^\star T - \hat{\mathcal{R}}_t^{\pi_t}(\boldsymbol{s}, T)}.
\end{aligned}$$

$\hat{\mathcal{R}}_s^{\pi_s}(\boldsymbol{s}, T)$ and $\hat{\mathcal{R}}_t^{\pi_t}(\boldsymbol{s}, T)$ can be bounded by the Empirical Bernstein bound [Audibert *et al.*, 2007]. With a probability $1 - \delta$, we have

$$\left| \hat{\mathcal{R}}_s^{\pi_s}(\boldsymbol{s}, T) - \mathbb{E}_{\pi_s}\left[\sum_{t=0}^{T} \mathcal{R}_s(\boldsymbol{s}_t, \boldsymbol{a}_t)\right] \right| \leq \epsilon_1,$$

with $\epsilon_1 = \bar{\sigma}\sqrt{\frac{2\log(3/\delta)}{n_s}} + \frac{6R_{max}\log(3/\delta)}{n_s}$, $\bar{\sigma} = \sqrt{1/n_s\sum_{i=1}^{n_s}(R_i - \bar{R})^2}$ is the standard deviation of the sample , we derive

$$\frac{\boldsymbol{\lambda}^\star T + d_t^s(\pi_s) - C_2}{\boldsymbol{\lambda}^\star T - C_4} \leq \hat{\rho}(\pi_s, \pi_t, T) \leq \frac{\boldsymbol{\lambda}^\star T + d_t^s(\pi_s) - C_1}{\boldsymbol{\lambda}^\star T - C_3} \tag{5}$$

with C_1, C_2, C_3, and C_4 are constants. Consequently, for negative transfer:

$$\hat{\rho}(\pi_s, \pi_t, T) \geq \frac{\boldsymbol{\lambda}^\star T + d_t^s(\pi_s) - C_2}{\boldsymbol{\lambda}^\star T - C_4} > 1.$$

Then, assuming enough samples, negative transfer occurs if:

$$d_t^s(\pi_s) > \left\{ \mathbb{E}_{\pi_s}\left[\sum_{t=0}^{T}\mathcal{R}_s(\boldsymbol{s}_t, \boldsymbol{a}_t)\right] - \mathbb{E}_{\pi_t}\left[\sum_{t=0}^{T}\mathcal{R}_t(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]\right\} \tag{6}$$

The condition sheds light on the negative transfer in the sense of metric notation and provides a formal way to determine negative transfer. First, if the condition in Eq. 6 holds after evaluation, researchers should avoid the source policy π_s to the target tasks since it may cause negative transfer. Second, if the researchers have enough expert knowledge about their working domain and transfer information, usually they can avoid this evaluation phase in practice. In short, Eq. 6 provides a formal way to understand negative transfer and justify the intuition (adopt high quality source knowledge and avoid bad teachers) in the transfer practice.

6 Experimental Results

Given the above theoretical successes, this section provides empirical validation on three domains:

Combination Lock: We use the domain described in Figure 2 which is a variation from [Whitehead, 1991]. The experimental setting follows the caption description.

Grid World is an RL benchmark in which an agent has to navigate an $m \times m$ grid world with the goal of reaching a goal state. We employ an 11×11 grid world with a four room layout as introduced in Sutton and Barto [1998]. The agent begins in the lower left corner of the map and navigates to the goal state being the upper right corner. To navigate, the agent has access (in each cell) to four actions transitioning it to the: north, south, west and east. Applying an action, it then transitions in that direction with a probability of 0.8 and in the other three with a probability of 0.2. In case the direction is blocked, the agent stays in the same state. Finally, the agent receives a reward of 0 once reaching the goal state and a reward of -1 in all others.

Block Dude is a game where an agent again navigates a maze to reach a goal state. Reaching the goal directly is impossible due to the presence of blocks restricting its movement. The agent, however, can move to the left, right, and upwards. To reach the goal state, it needs to pick-up blocks and relocate them in correct positions. We use the default level 1 BURLAP [MacGlashan, 2014] in which there are two blocks and 3×25 maze. The agent receives a reward of $+1$ in the goal state and a reward of -1 in all other states.

6.1 Experimental Setup & Results

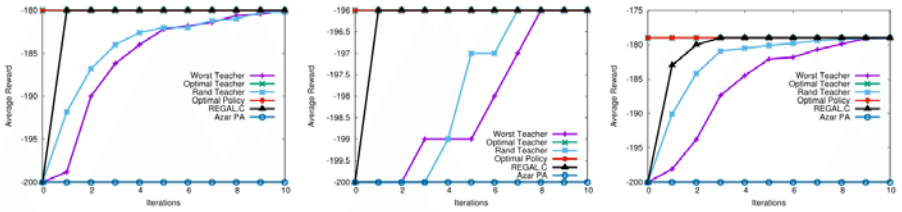
To construct the grand teacher, we set the total number of teachers $k = 10$. For each teacher, the budget, $\boldsymbol{b}_i$, is set to the total number of states. In Algorithm 3, the maximum number of iterations and the size of each dataset, $\mathcal{D}_i$, were set to 10 and 200, respectively. Values of $p^i = 0.5^i$ for $i = 1, \ldots, 10$ were used to determine $\boldsymbol{\beta}_i = p^i$. For Algorithm 1, the confidence $\boldsymbol{\delta}$ was set to 0.8 and H to 1000. The optimal gain $\lambda^\star$ and the optimal bias vector $\boldsymbol{h}$ can be approximated using the value function $\mathcal{V}$ [Puterman, 2005]. Let $\mathcal{V}^l$ be the value function at iteration l, $l = 0, 1, \ldots$. The optimal gain $\boldsymbol{\lambda}^\star \approx \text{sp}(\mathcal{V}^{l+1} - \mathcal{V}^l)$, where $\text{sp}(\mathcal{V}^l) = \max_{\boldsymbol{s}\in\mathcal{S}}\mathcal{V}^l(\boldsymbol{s}) - \min_{\boldsymbol{s}\in\mathcal{S}}\mathcal{V}^l(\boldsymbol{s})$ and the optimal bias vector $\boldsymbol{h}^\star \approx \mathcal{V}^l - l\boldsymbol{\lambda}^l$, when l is large enough. To smooth the natural variance in the student's performance, each learning curve is averaged over 10 independent trials of student learning. To better evaluate our method, we adopt six experimental settings by considering different teachers and learning algorithms. For teachers, we consider three forms. The first, referred to as "optimal teacher" provides optimal actions and is used by the grand teachers. The second, referred to as "worst teacher" advices the student to take actions with the lowest Q-values, while the last randomly selects action suggestions from the set of allowed moves. We also compare our method to REGAL.C (no advice), optimal policy (without learning), and Azar's method [Azar *et al.*, 2013]. Please note that Azar's method can not converge to the optimal policy and suffers loss as its performance is restricted by the teacher.

Performance, measured by the average reward, is reported in Figure 1. First, it is clear that given optimal teachers, our method exactly traces the optimal policy achieving a regret of 0. It is also important to note that in all three domains, even if the teacher was not optimal, and contrary to current techniques, our method is capable of acquiring optimal behavior. This is achievable as our method allows for learning within the multiple teacher framework.

7 Related Work on Transfer Learning

Few theoretical results on transfer and policy advice have been achieved. Closest to this work is that in Taylor *et al.* [2014], where the authors only provide empirical validations to their approach without drawing on any theoretical analysis. Given the theoretical derivations in this paper, we in fact note that the method [Taylor *et al.*, 2014] is a special case of ours considering only one-teacher advice models.

Another method considering advice under multiple teachers is that in Azar *et al.* [2013]. Azar *et al.* propose a method capable of selecting the best policy from a set of teacher policies and derive sub-linear regret of the form $\mathcal{O}(\sqrt{T})$ with T being the total number of rounds. One drawback of their method, however, is the assumption of a "good-enough" teacher which can guide the student to optimality. Such a method may suffer huge regret if the overall quality of teacher policies is poor. It also can not obtain better policies than those of the teacher. Our algorithm remedies these problems by allowing agents to further improve, which gives them the opportunity to surpass the teacher's performance.



(a) Grid World (b) Combination Lock (c) Block Dude

Figure 1: Our method with optimal teacher has similar performance as the optimal policy. And the REGAL algorithm (no advice) outperform random teacher and worst teacher group which justifies that the poorer teachers do harm the learning. Azar's method depends on the quality of the teachers — when the teachers are very poor, the algorithm shows no learning.

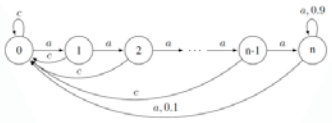


Figure 2: There are $n+1$ states in the MDP. The last state has only one action and the rest have two. The agent receives reward -1 for all actions, except when taking action a in state n, $\mathcal{R}(n,a)=1$. The agent stays in state n with probability 0.9 and goes to state 0 with probability 0.1. The optimal policy is to take action a in each state. Since there are $n+1$ states, the budget $\mathfrak{B}$ is at least $n+1$ to achieve zero regret.

Human advice is also a good source of policy advice. Usually, this method adopts the human advice as the teacher's policy to improve the learning performance. However, these works focus on empirical validations [Cakmak and Lopes, 2012; Griffith *et al.*, 2013].

Probabilistic policy reuse is similar to our method in which the algorithm follows its own knowledge with probability $1-\epsilon$ and teacher's policy with probability ϵ [Fernández and Veloso, 2006]. However, ϵ is not decaying over time, making the algorithm divergent if teacher policies are not optimal. Cederborg *et al.* introduce a policy shaping algorithm using human teachers, but focus on providing rewards rather than action advice [Cederborg *et al.*, 2015]. Both of these works rely solely on empirical results.

Work on transfer for RL is also related to this paper, where we can consider policy advice as an instance of transferring from teachers to students [Lazaric, 2012]. Here, Ferrante *et al.*, for instance, propose a method to transfer high quality samples from source to target tasks using bi-simulation measures [Ferrante *et al.*, 2008]. Their method only transfers samples once, while our approach gradually provides advice to the student. Due to space constraints, we refer the reader to Taylor and Stone [2009] for a comprehensive survey.

Lifelong reinforcement learning has drawn significant attention to the transfer community recently. Brunskill and Li studied online discovery problems in a lifelong learning setting [Brunskill and Li, 2015]. Bou-Ammar *et al.* also studied such a problem and introduced constraints on the policy to compute "safe" policies [Bou-Ammar *et al.*, 2015]. Contrary to these works, in this paper, we focus on the single agent setting operating within one task.

Finally, Learning from Demonstration [Argall *et al.*, 2009] (LfD) is also related to our work, but LfD usually assumes that the expert is optimal and the student only tries to mimic the expert.

8 Conclusion and Future Work

In this paper, we formally defined the multi-teacher advice model and introduced a new algorithm which leverages teacher and student's own knowledge in the weakly communicating MDPs. We theoretically analyzed our algorithm and showed, for the first time, that the agent can achieve optimality even when starting from non-optimal teachers. Our results provide a theoretical justification for the intuition that "bad" teachers can hurt the learning process of the student. Also, we formally established the condition of negative transfer, shedding light on future transfer learning research, where for example, researchers can choose "good teachers" based on the Eq 6 and avoid negative transfer with prior expert knowledge.

In future, we plan on adopting other online reinforcement learning algorithms (e.g., REGAL.D [Bartlett and Tewari, 2009], R-max [Brafman and Tennenholtz, 2003], or E^3 [Kearns and Singh, 2002]) to replace REGAL.C. We will provide better methods to construct the "grand-teacher" without exploring the whole MDP. Also, extensions to large-scale MDPs may be an interesting direction for future research as well.

9 Acknowledgements

This research has taken place in part at the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL research is supported in part by grants AFRL FA8750-14-1-0069, AFRL FA8750-14-1-0070, NSF IIS-1149917, NSF IIS- 1319412, USDA 2014-67021-22174, and a Google Research Award.

References

[Argall *et al.*, 2009] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[Audibert *et al.*, 2007] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Tuning bandit algorithms in stochastic environments. In *Algorithmic Learning Theory*, pages 150–165. Springer, 2007.

[Auer *et al.*, 2009] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in neural information processing systems*, pages 89–96, 2009.

[Azar *et al.*, 2013] Mohammad Gheshlaghi Azar, Alessandro Lazaric, Brunskill Emma, et al. Regret bounds for reinforcement learning with policy advice. In *ECML/PKDD-European Conference on Machine Learning and Principles and practice of knowledge Discovery in Database*, 2013.

[Bartlett and Tewari, 2009] Peter L Bartlett and Ambuj Tewari. Regal: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 35–42. AUAI Press, 2009.

[Bou-Ammar *et al.*, 2015] Haitham Bou-Ammar, Rasul Tutunov, and Eric Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. *CoRR*, abs/1505.05798, 2015.

[Brafman and Tennenholtz, 2003] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

[Brunskill and Li, 2015] Emma Brunskill and Lihong Li. The online discovery problem and its application to lifelong reinforcement learning. *CoRR*, abs/1506.03379, 2015.

[Cakmak and Lopes, 2012] Maya Cakmak and Manuel Lopes. Algorithmic and human teaching of sequential decision tasks. In *AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012.

[Cederborg *et al.*, 2015] Thomas Cederborg, Ishaan Grover, Charles L. Isbell, and Andrea L Thomaz. Policy Shaping With Human Teachers. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[Erez and Smart, 2008] Tom Erez and William D Smart. What does shaping mean for computational reinforcement learning? In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, pages 215–219. IEEE, 2008.

[Fernández and Veloso, 2006] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727. ACM, 2006.

[Ferrante *et al.*, 2008] Eliseo Ferrante, Alessandro Lazaric, and Marcello Restelli. Transfer of task representation in reinforcement learning using policy-based proto-value functions. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1329–1332. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[Griffith *et al.*, 2013] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2625–2633, 2013.

[Kearns and Singh, 2002] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[Lazaric, 2012] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[MacGlashan, 2014] James MacGlashan. The Brown-UMBC Reinforcement Learning and Planning (BURLAP) http://burlap.cs.brown.edu/index.html, 2014.

[Puterman, 2005] Martin L Puterman. Markov decision processes: Discrete stochastic dynamic programming (wiley series in probability and statistics). 2005.

[Ross *et al.*, 2010] Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686*, 2010.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[Taylor and Stone, 2009] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.

[Taylor *et al.*, 2014] Matthew E. Taylor, Nicholas Carboni, Anestis Fachantidis, Ioannis Vlahavas, and Lisa Torrey. Reinforcement learning agents providing advice in complex video games. *Connection Science*, 26(1):45–63, 2014.

[Whitehead, 1991] Steven D Whitehead. Complexity and cooperation in q-learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 363–367, 1991.

[Zimmer *et al.*, 2014] Matthieu Zimmer, Paolo Viappiani, and Paul Weng. Teacher-Student Framework: a Reinforcement Learning Approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

Towards Integrating Real-Time Crowd Advice with Reinforcement Learning

Gabriel V. de la Cruz Jr.,
Bei Peng
School of EECS
Washington State University
gabriel.delacruz@wsu.edu,
bei.peng@wsu.edu

Walter S. Lasecki
Computer Science Department
University of Rochester
wlasecki@cs.rochester.edu

Matthew E. Taylor
School of EECS
Washington State University
taylorm@eecs.wsu.edu

ABSTRACT

Reinforcement learning is a powerful machine learning paradigm that allows agents to autonomously learn to maximize a scalar reward. However, it often suffers from poor initial performance and long learning times. This paper discusses how collecting on-line human feedback, both in real time and *post hoc*, can potentially improve the performance of such learning systems. We use the game Pac-Man to simulate a navigation setting and show that workers are able to accurately identify both when a sub-optimal action is executed, and what action should have been performed instead. Demonstrating that the crowd is capable of generating this input, and discussing the types of errors that occur, serves as a critical first step in designing systems that use this real-time feedback to improve systems' learning performance on-the-fly.

INTRODUCTION

Reinforcement learning [7] is a very flexible, robust approach to solving problems. However, early in the training process much of the problem space is unexplored, often resulting in poor performance because reasonable policies are only discovered after a considerable amount of trial-and-error. In this paper, we propose the idea of using on-demand human intelligence, available via crowdsourcing platforms such as Amazon Mechanical Turk, to provide immediate feedback to reinforcement learning systems based on the intuition and experience of the human observer.

To test whether crowd workers are able to accurately provide such advice, we perform a set of experiments that measure the crowd's ability to generate just-in-time warnings to an agent playing Pac-Man. First, we establish that the crowd can collectively identify the correct point at which an error occurs with over 91% accuracy. Second, we demonstrate that not only can this mistake identification be done in real time with a mean latency of just 0.39 seconds, but also that workers are able to identify what the optimal move *would* have been been.

IUI'15 Companion, Mar 29 - Apr 01, 2015, Atlanta, GA, USA
ACM 978-1-4503-3308-5/15/03.
http://dx.doi.org/10.1145/2732158.2732180

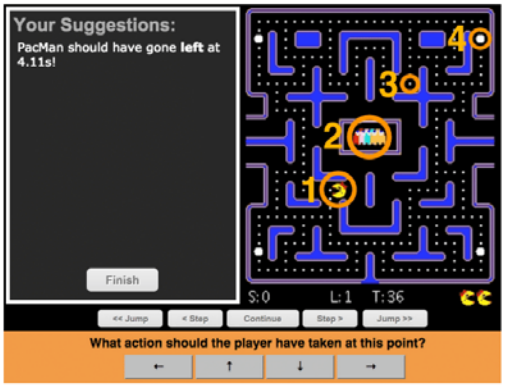


Figure 1: This screenshot shows the web interface of the user study with game layout, and components of the Pac-Man game: 1) Pac-Man, 2) 4 Ghosts, 3) Pills, and 4) Power Pills.

Third, we compare the crowd's performance in this real-time setting with an offline "review" setting where game playback can be controlled and replayed. In this setting, mistakes can be better estimated, with a mean distance from the correct position of just 0.15 seconds.

This work is the first research to establish the crowd's ability to react to mistakes made by an intelligent agent in real time, and provide accurate guidance on a preferred alternative action. Our work informs the design of future systems that use human intelligence to guide untrained systems through the learning process, without limiting systems to only learn from their mistakes far after they make them.

The contributions of this paper are to:

- Present the idea that on-line crowds can provide very accurate assistance to learning agents by using real-time data.
- Demonstrate that crowd workers can respond quickly and accurately enough to provide just-in-time feedback.
- Show that workers can also improve their accuracy in *post hoc* review settings for use in future situations.

BACKGROUND AND RELATED WORK

Reinforcement learning has a history of succeeding on difficult problems with little information. This paper leverages the on-policy learning algorithm Sarsa [7]. A Sarsa agent learns to estimate Q-values, representing the estimated total reward the agent would receive in a state s, execute action a, and then follows the current policy until the episode is terminated. Over time, this type of temporal difference learning allows the agent to learn a (near) optimal policy that collects as much reward as possible, in expectation.

While autonomous methods like Sarsa have many empirical successes and sound theoretical underpinnings, if a human is available to provide useful information, the agent is often able to learn much faster. For example, a user could make judgements about the agent's performance by providing human reward [3, 6] or providing demonstrations. Finally, when a human can temporarily control the agent to perform the correct behavior, learning from demonstration techniques [1].

Most related to this paper is existing work done on crowdsourcing control and recognition tasks. Human control of robots has been previously explored in the context of a robot Ouija board and navigation setting [2]. The Robot Management System [10] (RMS) also uses on-line contributors to crowdsource human-robot interaction studies. RMS used groups of participants working from their home computers to practice controlling a robot using camera views and a web-based control interface.

Legion [4] explored using crowd workers to collaboratively control a robot in real time. This was the first work to show that on-demand human intelligence could be used to control a robot when an automatic system is unable to proceed. Legion:AR [5] showed that an active learning approach could be used in an activity recognition setting to call on crowd support for an action-labeling task only when needed. In both systems, low-latency responses were achieved by keeping the crowd continuously engaged with a task for a period of time. However, complete crowd control does not let the system effectively evaluate its own policy. In this work, we explore if and how we can use real-time crowds in an advisory role, without needing the crowd to directly control the Pac-Man avatar, while still maintaining exceptional response speeds.

EXPERIMENTAL DESIGN

Our Pac-Man agent (see Figure 1) used Sarsa to learn a near-optimal policy to win the game while earning as many points as possible using an existing open learning implementation [8]. Due to the large state space, the agent uses seven high-level features for function approximation to learn a continuous Q-value function. Pac-Man code is available from `http://www.eecs.wsu.edu/~taylorm/l3PacMan.zip`.

To generate the videos used in the user study, we recorded Pac-Man being controlled by a human who intentionally made different types of mistakes. Then, we picked 10–14 seconds which contained one (and only one) mistake. Q-values for the agent's trajectory were also recorded, confirming that the human-created mistakes had lower Q-values than the "correct" action. We create four videos where each contained a mistake: Video 1) moving so that Pac-Man is trapped by one or more ghosts, Video 2) not moving towards an edible ghost after eating a power pill, Video 3) taking an empty path instead of going for pills when they are no risk, and Video 4) not going for all edible ghosts that are within close range.

To study the hypothesis that crowd workers can provide information useful to reinforcement learning agents, we consider four settings. First, a video of Pac-Man is played only once (*real-time*) or the worker can view it multiple times (*review*). Second, the worker may be asked to identify the time at which the mistake is made (*Mistake Identification*), or asked to identify both the mistake time as well as suggest the optimal action (*Action Suggestion*).

We want to measure the performance of users in identifying the point at which a mistake is made and suggesting optimal action Pac-Man should have executed. To evaluate worker actions, we can compare to recorded Q-values.

USER STUDIES

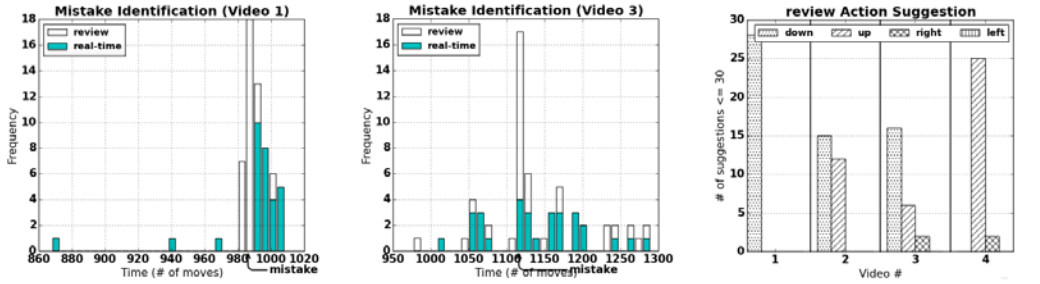
Workers were first shown instructions describing the task, as well as the rules of Pac-Man. During a preliminary test of the web interface, we found that workers would sometimes identify mistakes *before* the sub-optimal action was executed, anticipating the mistake. We provided explicit instructions to workers to encourage them to identify the exact time at which a mistake was made. Workers were then directed to a tutorial which asked them to complete an example task using the marking interface. After the tutorial, workers will watch a new video and must press a button (see Figure 1) as soon as they observe a mistake.

We recruited crowd workers from Amazon Mechanical Turk (AMT) for our experiments. While AMT provides immediately, programmatic access to crowds, it also poses a number of challenges, including that workers: 1) are unlikely to be experts, 2) may not take the task seriously and not read the instructions, and 3) may intentionally select incorrect times/actions. Our methods need to be robust to these challenges, unlike in Learning from Demonstration, where demonstrations are typically assumed to be optimal.

16 Human Intelligence Tasks (HITs) on AMT encompassed our four different types of experiments. Each experiment was tested with 4 different videos. We collected data from 30 unique workers per HIT and every worker was paid 25 cents.

RESULT ANALYSIS

This section presents the results of our study in three parts. First, we establish that the crowd can identify the mistake with high accuracy. Second, we demonstrate that not only can Mistake Identification be done in real-time but that workers can also successfully identify what the optimal "correct" move would have been. Third, we compare the crowd's performance in the real-time setting with offline "review" setting and show that if additional time is available, even more accurate performance can be achieved.



(a) A histogram of workers' suggestions using Video 1 (mistake time: 986).

(b) A histogram of workers' suggestions using Video 3 (mistake time: 1116).

(c) Number of times each action was suggested by workers (Optimal: 1–down, 2–down, 3–down, 4–up).

Figure 2: Selected exemplar results from our 16 Amazon Mechanical Turk experiments.

Mistake Identification

Our performance measure is based on how many workers can correctly identify and suggest a time that is close to the correct mistake time. Histograms provide a visual representation of the accuracy of workers in different settings. The mistake times are reported as game move numbers, which are 986, 1809, 1116 and 334, for Videos 1–4, respectively. These video clips are 10 to 14 seconds long, corresponding to 250–350 total game moves, and the mistakes located roughly three quarters of the way through the clip. However, because Pac-Man moves continually, it is difficult for workers to identify the exact frame when the mistake was executed.

To quantify how accurate the workers were, we calculated the difference between the actual mistake time and the identified mistake time, where zero corresponds to a perfect answer. We selected a threshold of 30 actions, roughly 1 second, so that any answer within ± 1 second will be counted as correctly identifying the mistake. Figure 2(a) shows the distribution of workers' answers where responses within the 956–1016 moves range are considered to be correct, showing only two errant responses.

To compute the mean difference between the time reported by a worker and the real error time μ_{diff}, we use: $\mu_{diff}(AMT_k) = \frac{\sum_{i=1}^{n} |t_{w_i} - t_m|}{n}$, where k is the group number, n is the total number of workers per group that are within the threshold, t_{w_i} is the ith worker's suggested time, and t_m is the correct mistake time. The standard deviation is also computed using: $\sigma_{diff}(AMT_k) = \sqrt{\frac{\sum_{i=1}^{n} (\mu_{diff}(AMT_k) - |t_{w_i} - t_m|)^2}{n}}$, where a low value indicates suggestions are tightly clustered.

To establish that workers can correctly identify where and when mistakes occur in our game, we count the number of people who correctly identified the mistake. Video 1's review setting collective percentage of correct events has the highest over all four videos with 98.3%. This is followed by Videos 4 and 2, with 88.0% and 86.6% respectively. And Video 3 has the lowest accuracy with 68.4%. This observed high percentage of correct events from the three videos suggests that the crowd can identify a mistake in many cases.

It is also important to point out that there are instances in which the mistakes are more subtle, making it harder to identify. Video 3 has the lowest accuracy, and the Mistake Identification experiment has the least percentage of correct answers at 56.7%. However, the sparsity of the data as shown in Figure 2(b) suggests that the mistake was harder to find.

In summary, these results established that, in most cases, workers can identify a mistake in the Pac-Man game with an overall accuracy for review Mistake Identification at 80% and an accuracy of 91% for review Action Suggestion.

Optimal Action Identification

Given that workers can correctly identify mistakes, we next consider whether they can also accurately provide the action that should have been taken. To do this, we first have to verify that majority of the workers within the threshold suggested the same action, and second, the suggested action has the maximum Q-value in the recorded video's game state.

Figure 2(c) shows that all workers' suggested actions that are within the 30-move threshold, in both the real-time and review cases, meaning that a majority of workers do suggest similar actions.

Knowing that the crowd reaches consensus on a single action, we can now compare the crowd's advice to the recorded Q-values of the game to verify if it is the correct (near-optimal) action. The maximum Q-value of the 4 possible Pac-Man actions determines what action Pac-Man should perform. In Video 1, a step before 986 moves should suggest the Q-values for the next move. At move 985, the Q-values are: $up = 1729$, $right = 1621$, $down = 1768$, and $left = 1621$. In the human-controlled game in Video 1, Pac-Man went right at this time when it should have gone down (the maximum Q-value). And as shown in Figure 2(c), workers did suggest for Pac-Man should move downward in Video 1. Similar in the other three videos demonstrate that workers can identify that a mistake has been made but as well as provide an advice that is useful and near-optimal.

Real-time vs. Review

We expected the real-time setting to be considerably harder than review setting. This assumption can be verified by considering the mean difference for each setting — the average mean difference for real-time setting is 9.1 moves (≈ 0.36 seconds) while review case is of 4.5 moves (≈ 0.18 seconds). The lower mean difference in review experiments shows that if additional time is available, even closer estimates of the point of the mistake can be gathered.

We performed a 4×2 Between Subjects Factorial ANOVA test of all Action Suggestion experiments shows that the difference of suggested mistake time by workers between subjects real-time and review setting was statistically significant ($F = 5.10, p < .05, \eta^2 = .023$). This difference between real-time and review setting in all Mistake Identification experiments is also significant ($F = 5.02, p < .05, \eta^2 = .022$). This indicates that the different mistakes in each video can also affect worker's ability to identify them.

Interestingly, there is only a small difference between the mean difference of real-time and review setting in Mistake Identification for Video 2. This indicates that the mistake in Video 2 was harder for workers to identify than the mistakes in the other three videos.

It is notable here that the average of mean differences in the real-time setting for Mistake Identification results to 9.8 moves (≈ 0.39 seconds), and with Action Suggestion at 8.8 moves (≈ 0.35 seconds), which are both very close to the human response for tasks with no high-level reasoning needed (e.g., clicking a button in response to a visual stimulus). This suggests that crowd advice for tasks, such as navigation, can be collected nearly as fast as people can physically respond. This quickly-available input can, in turn, be used to improve real-time learning of virtual and physical agents.

FUTURE WORK

Future work will focus on developing learning algorithms that to leverage the unique strengths of human input on-the-fly without being detrimentally affected by incorrect advice. Although others [9] have incorporated advice from multiple demonstrators in past work, errors from crowdsourced workers are a unique challenges and opportunities to scale these systems. Furthermore, we plan to continue to improve our interfaces to further reduce the latency of worker responses. One potential method to do this is to leverage workers' ability to predict when mistakes might be made, which we initially observed, to collectively decrease latency below the best after-the-fact response speed possible. We are also interested in studying how the number of examples during the tutorial affects participants' accuracy. Finally, we are interested in eliciting a confidence measure from workers, potentially allowing us to weight different pieces of advice.

CONCLUSION

Reinforcement learning algorithms often suffer from poor early-stage performance since agents have to experience considerable amount of trial-and-error before learning an effective policy. Our approach uses real-time crowds to provide immediate assistance to the learning agent to help improve its performance. We ran a set of user studies to show that crowd workers from Amazon Mechanical Turk can respond quickly and accurately enough to provide just-in-time feedback to an agent playing Pac-Man. We show that workers can correctly identify the point at which a mistake is made by Pac-Man and the optimal action Pac-Man should have executed. We also showed that higher performance could be achieved by workers in *post hoc* review settings.

Our results demonstrated that 1) crowd workers are able to accurately choose the mistake time in real-time with a mean latency of just 0.39s, and 2) latency does not increase if workers must also suggest an action. By leveraging the crowd, we present an effective, scalable means of providing during-task assistance to learning agents.

ACKNOWLEDGEMENTS

The authors would like to thank our anonymous reviewers and Leah Zulas for helpful comments and suggestions. This work was funded in part by NSF IIS-1319412, and a Microsoft Research Ph.D. Fellowship.

REFERENCES

1. Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robot. Auton. Syst. 57*, 5 (May 2009), 469–483.
2. Goldberg, K., Chen, B., Solomon, R., Bui, S., Farzin, B., Heitler, J., Poon, D., and Smith, G. Collaborative teleoperation via the internet. In *Proc. of ICRA* (2000).
3. Knox, W. B., and Stone, P. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proc. of AAMAS* (2010).
4. Lasecki, W. S., Murray, K. I., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *Proc. of UIST* (2011).
5. Lasecki, W. S., Song, Y. C., Kautz, H., and Bigham, J. P. Real-time crowd labeling for deployable activity recognition. In *Proc. of CSCW* (2013).
6. Loftin, R., Peng, B., MacGlashan, J., Littman, M. L., Taylor, M. E., Huang, J., and Roberts, D. L. A strategy-aware technique for learning behaviors from discrete human feedback. In *Proc. of AAAI* (2014).
7. Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*, vol. 28. MIT press, 1998.
8. Taylor, M. E., Carboni, N., Fachantidis, A., Vlahavas, I., and Torrey, L. Reinforcement learning agents providing advice in complex video games. *Connection Science 26*, 1 (2014), 45–63.
9. Taylor, M. E., Suay, H. B., and Chernova, S. Integrating reinforcement learning with human demonstrations of varying ability. In *Proc. of AAMAS* (2011).
10. Toris, R., Kent, D., and Chernova, S. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction 3*, 2 (2014), 25–49.

List of Symbols, Abbreviations, and Acronyms

AAAI: AAAI Conference on Artificial Intelligence
ELLA: Efficient Lifelong Learning Algorithm
HIT: Human Intelligence Task
ICML: International Conference on Machine Learning
IJCAI: International Joint Conference on Artificial Intelligence
IROS: International Conference on Intelligent Robots and Systems
IUI: Association of Computing Machinery Conference on Intelligent User Interfaces
MDP: Markov Decision Process
MTL: Multi-task Learning
PG-ELLA: Policy Gradient ELLA
Q-Learning: Q-value Learning, a type of TD-Learning
RL: reinforcement learning
SDM: sequential decision-making
TD-Learning: Temporal Difference Learning, a type of RL algorithm